

NPS-53Fe76051

NAVAL POSTGRADUATE SCHOOL

Monterey, California



A PROGRAM FOR THE NUMERICAL SOLUTION OF LARGE SPARSE
SYSTEMS OF ALGEBRAIC AND IMPLICITLY DEFINED STIFF
DIFFERENTIAL EQUATIONS

by

Richard Franke

May 1976

Technical Report For Period

October 1975 - April 1976

Approved for public release; distribution unlimited

FEDDOCS
D 208.14/2:
NPS-53FE76051

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral Isham Linder
Superintendent

Jack R. Borsting
Provost

ABSTRACT

This report documents a program for the numerical solution of large sparse systems of algebraic and implicitly defined stiff differential equations. The principal use is intended to be the solution of differential equations arising from time dependent partial differential equations when the finite element method is used to discretize the space domain. The use of compact matrix storage techniques and iteration for the solution of the quasi-Newton iterates in Gear's method makes the program extremely efficient both in terms of storage requirements and execution times.

Reproduction of all or part of this report is authorized.

This report was prepared by:

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NPS-53Fe76051	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Program for the Numerical Solution of Large Sparse Systems of Algebraic and Implicitly Defined Stiff Differential Equations		5. TYPE OF REPORT & PERIOD COVERED Technical Report Oct. 1975 - April 1976
7. AUTHOR(s) Richard Franke		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Foundation Research Program Naval Postgraduate School Monterey, California 93940		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS Chief of Naval Research Arlington, Virginia 22217		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61152N;RR 00-01-10; N0001476WR60052
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE May 1976
		13. NUMBER OF PAGES
		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Gear's Method Differential-Algebraic systems Stiff equations Sparse matrices Finite element method		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report documents a program for the numerical solution of large sparse systems of algebraic and implicitly defined stiff differential equations. The principal use is intended to be the solution of differential equations arising from time dependent partial differential equations when the finite element method is used to discretize the space domain. The use of compact matrix storage techniques and iteration for the solution of the quasi-Newton iterates in Gear's method makes the program extremely efficient both in terms		

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

of storage requirements and execution times.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

1.0 Introduction

This report documents a program for the solution of algebraic and implicitly defined stiff differential equations. We were particularly interested in solving very large systems of differential equations arising from partial differential equations where the finite element method has been applied in the space variables.

Our original goal was to use a compact storage scheme for the large matrices involved and to use iteration to solve the linear algebraic systems which occur. However, the resulting program is easily adapted to different applications through user modifications accomplished by replacement of one or two relatively simple subroutines. Thus the program is a powerful one which can be used in a variety of applications. Four examples, illustrating different matrix storage techniques and different linear equation solvers are given in the appendix. Other storage schemes and solution methods, e.g. symmetric Jacobian with Cholesky decomposition, are relatively simple to implement.

In Section 2 a brief review of the integration scheme is given. In Section 3 a discussion of differences between this program and the one from which it was adapted is given. Section 4 is devoted to a discussion of information concerning the use of this package.

2.0 Theoretical Background

Consider the system of $N = m + \ell$ differential and algebraic equations in $y_1, \dots, y_m, v_1, \dots, v_\ell$,

$$(1) \quad F(y, \dot{y}, t) + P(t) V = 0 ,$$

with all or some of the initial values $y_1(t_0), \dots, y_m(t_0), v_1(t_0), \dots, v_\ell(t_0)$ specified. Enough of the above values must be given in order to determine the remaining values and initial values for any of the derivatives, $\dot{y}_1, \dots, \dot{y}_m$ which appears in equation (1). In equation (1), $P(t)$ is an $N \times \ell$ matrix, F is a vector with N components, and V is a vector.

The program documented here is a modification of a program due to Brown and Gear [1]. The method of solution is a modification of Gear's method for stiff differential equations [2]. The application to differential algebraic systems was given by Gear [3]. We will briefly describe the method here for completeness, and refer the reader to the references for more details.

Suppose that approximate solution values are known at a number of equally spaced points, $t_{n-1}, t_{n-2}, \dots, t_{n-k}$, and are represented by $y^{(n-1)}, \dots, y^{(n-k)}$ respectively. Let $V(t_{n-1})$ be represented by $V^{(n-1)}$. Use of a backward differentiation formula gives

$$h \dot{y}^{(n)} = \frac{1}{\beta_0} (\alpha_0 y^{(n)} + \alpha_1 y^{(n-1)} + \dots + \alpha_k y^{(n-k)}) ,$$

where $h = t_{i+1} - t_i$. The coefficients α_i and β_0 are from Gear [2], p. 217. Substitution of this into (1) gives

$$(2) \quad F(y^{(n)}, -\frac{\alpha_0}{\beta_0 h} y^{(n)} + \sum_n, \dot{t}_n) + P(t_n) V^{(n)} = 0$$

as the equation which $y^{(n)}$ and $V^{(n)}$ must satisfy. In equation (2),

$$\sum_n = \frac{1}{\beta_0 h} (\alpha_1 y^{(n-1)} + \dots + \alpha_k y^{(n-k)}) .$$

In general, equation (2) represents a system of nonlinear equations for $y^{(n)}$ and $v^{(n)}$. The method used for solving this system of algebraic equations is a variant of Newton's method. The initial guess, $y^{(n),0}$, is obtained by polynomial extrapolation using a Hermite polynomial interpolating the known values $\dot{y}^{(n-1)}$, $y^{(n-1)}$, ..., $y^{(n-k)}$. Thus the predicted values has the form

$$(3) \quad y^{(n),0} = h \bar{\beta}_1 \dot{y}^{(n-1)} + \bar{\alpha}_1 y^{(n-1)} + \dots + \bar{\alpha}_{n-k} y^{(n-k)}.$$

The application of Newton's Method to equation (2) then yields the corrector equation

$$(4) \quad J \cdot \begin{pmatrix} y^{(n),i+1} - y^{(n),i} \\ v^{(n),i+1} - v^{(n),i} \end{pmatrix} = -F(y^{(n),i}, -\frac{\alpha_0}{\beta_0 h} y^{(n),i}, t_n) + P(t_n) v^{(n),i},$$

where J is the Jacobian matrix,

$$(5) \quad J = \begin{pmatrix} \frac{\partial F}{\partial y} - \frac{\alpha_0}{\beta_0 h} \frac{\partial F}{\partial \dot{y}} & P \\ \vdots & \vdots \end{pmatrix}.$$

Gear shows that the initial guess for $v^{(n)}$ is not important, and thus $v^{(n-1)}$ is used. Up to three iterations are performed on the corrector equation. The matrix J is not evaluated at each iteration, nor even at each timestep. J is evaluated whenever (i) the timestep or order is changed, or (ii) the corrector iteration fails to converge in three iterations.

If the corrector iteration fails to converge, the J matrix is evaluated, unless it had already been evaluated at the current time.

If it has been evaluated at the current time, the timestep is reduced by a factor of 4. In either case the step is then retired.

If the corrector iteration converges, the local error is estimated, based on the fact that the local error is approximately proportional to the difference between the predicted and corrected values of $y^{(n)}$. For this purpose a relative error tolerance is used for large solutions and an absolute error tolerance for small solutions. The root-mean-square norm (Euclidean norm divided by the square root of the number of components) is used for the vector with components e_i / y_{\max_i} , where e_i is the estimated local error in $y_i^{(n)}$ and $y_{\max_i} = \max_{0 \leq k \leq n} (|y_i^{(k)}|, 1)$. If the error is larger than that specified by the user, an acceptable timestep is estimated for the current order or order one lower, and the step repeated. Up to three such failures are permitted, after which an attempt is made to start over with a first order method.

When using a method of order q , the program takes at least $q + 1$ steps before changing the timestep. Changes in timestep are preceded by calculation of the predicted timestep at current order and order one higher and one lower. If the timestep can be increased by more than 10%, the order corresponding to the largest possible timestep is used. If the timestep cannot be increased by at least 10%, the current order and stepsize are retained for at least 10 more steps.

After each step a test is made to determine whether time has advanced to or beyond the input end time. Control is returned to the calling program if it has.

At the initial call, no history of the solution is available, so the program must begin with a first order method, taking two such steps in accordance with the above description. The timestep must be suitably small, again in accordance with the above. At the point the program can begin to increase the order of the method and the timestep. Because the Jacobian matrix must be generated whenever the timestep is changed, it is not efficient to try too large a timestep initially. Because the program rapidly finds the best order and timestep, it is relatively cheap to underestimate the initial timestep compared to the cost of overestimating it.

3.0 Differences compared with DFASUB

The principal differences between the SDESOL/LDASUB package and DFASUB, and the reasons for incorporating them are as follows.

(i) The main goal of this revision was to generate a program which could solve very large sparse systems of differential equations efficiently, both in terms of storage requirements and execution time. We are particularly interested in the solution of ordinary differential equations arising from partial differential equations where the finite element method has been used to discretize the problem in space.

Large sparse problems require at least a different system of storing the Jacobian matrix and possibly the use of iteration to solve for the quasi-Newton iterates in equation (2.4). Two such subroutine packages, to be used with the basic subroutines, have been provided. Another package using standard elimination techniques is also provided and is convenient for smaller systems of equations. Use of any of these

options requires the user to supply a subroutine to evaluate his form of the equations (1), and for efficiency, a subroutine to explicitly evaluate the Jacobian. A subroutine to approximate a full Jacobian through numerical differencing has been provided. With the exception of a minor correction, this is the same as given in [1]. While use of this routine is convenient, it is inefficient and should be avoided for large systems. It is anticipated that the user can provide his own subroutine package, using his own storage scheme for the Jacobian, and with a suitable equation solver for the Newton iterates. There should be no need to disturb the basic package which carries out the time integration.

(ii) For user convenience, without a major rewrite of DFASUB, a driver routine, SDESOL, to be referenced by the user and which then communicates with LDASUB was written. The chief function of SDESOL is to set up a number of references to work storage areas required by LDASUB. In addition, some testing of parameters is accomplished, and a subroutine to calculate initial values of derivatives is called.

(iii) A subroutine to calculate initial values of derivatives, DERVAL, has been provided. The routine provided requires that the first m rows of $\frac{\partial F}{\partial \dot{y}}$ be nonsingular, which does not need to be true in the general case. For this reason, and others discussed in Section 4, the user may need to provide either his own version of DERVAL to evaluate the derivatives initially, or else he may supply initial values and a dummy version of DERVAL.

(iv) Other changes made in generating LDASUB from DFASUB were to simplify the code for the particular type of problem we wish to solve, while

others were to enhance the usability of the code. Some errors were also corrected, notably two errors in coefficients for the fifth and sixth order methods. DFASUB had the capability of computing various elements of the Jacobian at different times if they had different dependencies, with the possibility of inverting that part of the matrix at that time, if it could be done. This could result in increased efficiency in certain problems, at some expense in convenience, but for our purposes it was not considered useful, and was removed. Therefore only one call is made to evaluate the Jacobian. The Jacobian was evaluated at the beginning of each timestep in DFASUB, but this has been eliminated in LDASUB, in accordance with the description in Section 2. A subroutine, S2, was called from DFASUB to evaluate time dependent terms whenever time was changed. This is reasonable, since the function evaluation routine may be called several times at a given value of the independent variable. We have removed this, preferring to test for a new time in the routines where time dependent terms appear, then evaluating and storing them internally to that routine when necessary. This helps make the function evaluation more self contained, as well.

In DFASUB extra parameters in the calling sequence allowed the user to communicate constants to the function evaluation and Jacobian subroutines via DFASUB. We believe this is inefficient and confusing, and removed this capability, preferring to communicate from the main program to these subroutines via Common, or possibly through multiple entry points.

The norm used for error tests in DFASUB is the Euclidean norm. This has the undesirable property that for large systems the allowable

error criterion may be large. We therefore changed to the root-mean-square norm in LDASUB, which is simply the Euclidean norm divided by the square root of the number of components. One other change was made in the error tests. As noted in Section 2, the error vector has components e_i / y_{\max_i} , where e_i is the estimated local error in the i^{th} variable y_i , and $y_{\max_i} = \max_{0 \leq k \leq n} (|y_i^{(k)}|, 1)$. In DFASUB the maximum was taken only up to the previous timestep, $n-1$. This change was incorporated because some of the problems in which we were interested began with many components at zero, but which very rapidly became large, around 10^{12} or more. Without updating the value of y_{\max_i} , the size of the timestep was artificially kept extremely small in order to satisfy an unreasonable error tolerance. For this reason, the maximum value of the component was updated before the norm of the relative errors was computed. For problems where values range near to one, the modification will result in no appreciable change in performance.

The printout of counters, timestep, time, and values of the dependent variables was made an option through a parameter in the calling sequence. An additional value printed is the order of the method being used.

DFASUB incorporated the capability of terminating if a certain number of floating point overflows had occurred. This capability was removed from LDASUB.

The final modification to the program was the incorporation of a restart capability without having to begin again with a first order method. This was accomplished by adding two entry points to LDASUB. One, LDASAV, saves values internal to LDASUB, returning them to the main program, where they can be saved for the time at which the calculation

is to be resumed. At that time, another entry point, LDARST, restores those values internal to LDASUB, while other necessary values are restored through the calling sequence.

4.0 Subroutine Descriptions

The description of subroutines is divided into two subsections. The first deals with the basic integration routine and other subroutines which make up the core package, and which should not be modified by the casual user. The second deals with a set of supporting subroutines, at least one of which must be supplied by the user since it defines his system of equations. The others may be usable in the form given in one of the examples, or can be defined by the user to accomplish his desired implementation.

4.1 Basic Subroutine Package

4.1.1 Subroutine SDESOL

This routine is the only one which needs to be referenced by most users. It serves as a driver for the integration routine, LDASUB. SDESOL has a simpler calling sequence than LDASUB and relieves the user of having to set up a number of auxiliary storage arrays. In addition, the routine calls DERVAL to calculate the values of the derivatives on the initial call.

The calling sequence is

```
CALL SDESOL(Y,YL,T,TEND,NY,NL,M,JSKF,MAXDER,IPRT,H,HMIN,HMAX,RMSEPS,W)
```

where the parameters are defined as follows.

Y - Input and output. An array dimensioned (7,NY). On the initial call this array contains the initial values of the dependent variables y_i , $i=1, \dots, m$ in $Y(1,i)$. During execution of the program the approximate values of $\frac{d^j y_i}{dt^j} \cdot \frac{h^j}{j!}$ is stored in $Y(j+1,i)$. Here h is the current stepsize. These values must not be changed between returns to the calling program and subsequent entries to SDESOL. It is possible to interpolate for values of the dependent variables at times other than those calculated by using the formula $y_i(t+s) = \sum_{j=0}^q Y(j+1,i) \left(\frac{s}{h}\right)^j$, where q is the order formula currently in use, and is obtained as $q = \lfloor \text{JSKF}/10 \rfloor$.

YL - input and output. Array of linear variables, v_i , $i=1, \dots, \ell$. The user supplies initial values for these variables, and during execution it contains current values of the linear variables.

T - input and output. The user supplies the initial time, which is updated to current time during execution.

TEND -input. Time at which the integration is to end. This is the only parameter normally changed by the user between successive entries to SDESOL.

NY - input. The number of differential and nonlinear variables, m .

NL - input. The number of linear variables, ℓ . This may be zero.

M - input. The number of variables to be included in the local error test. The error test will be performed for the variables y_i , $i=1, \dots, M$. The M used is no greater than NY .

JSKF -input and output. An indicator: on input,

JSKF = 0 indicates that this is the initial call to SDESOL.

Initial values of the derivatives are calculated and auxiliary storage references are set up. This also indicates to subroutine LDASUB to initialize parameters and begin with a first order integration method.

JSKF > 0 indicates a continuation of a previous call to SDESOL .

JSKF = - 1 indicates a restart call. This is discussed further in Section 4.1.2.

JSKF < - 1 may result from the user neglecting to test for error returns from SDESOL. Because of this possibility, the run is terminated with an appropriate comment when JSKF < - 1 is input.

On output, JSKF normally is a two digit number, $\pm qp$. q is the order of the formula currently being used for the integration. p is an indicator determining the type of return. JSKF > 0 , $p = 1$ is the normal return. Note that SDESOL may be re-entered to continue the solution without changing JSKF. JSKF < 0 is an error return, with the value of p indicating the error, as follows.

$p = 1$ error test failure for $H \geq H_{MIN}$

$p = 3$ corrector failed to converge for $H > H_{MIN}$

$p = 4$ corrector failed to converge for a first order method

$p = 5$ error return from subroutine NUITSL

$p = 6$ error return from subroutine DERVAL

MAXDER-input. Maximum order method which should be used. The highest order possible is six.

IPRT- input. Print control indicator.

≤ 0 , no print from LDASUB

> 0 , at each step, print number of steps, number of Jacobian evaluations, current order being used, stepsize for next step, current time, and current values of the dependent variables.

H - input and output. On initial call it is an estimate of the timestep. During execution it is updated to the current value, and on return contains the stepsize to be tried for the next step. The input value need not be accurate. It is better to underestimate than to overestimate the initial value. The stepsize and order are varied to meet the local error tolerance specified. The user does not normally change the stepsize between entries to SDESOL.

HMIN- input. The minimum stepsize to be allowed.

HMAX- input. The maximum stepsize to be allowed.

RMSEPS-input. The error test constant. The values of the relative local errors must have root-mean-square norm less than RMSEPS.

W - an array of auxiliary storage required by LDASUB. This array must contain a total number of locations equal to the sum of (i) $13*NY + 5*NL$ for arrays used in LDASUB, (ii) storage for the Jacobian matrix, and (iii) any locations used in processing the Jacobian, e.g., scratch storage used by an equation solver.

4.1.2 Subroutine LDASUB

This subroutine is the basic integration routine and performs the process in essentially the same manner as subroutine DFASUB. A brief description is given in Section 2 and differences between this routine and DFASUB are outlined in Section 3. Parameters in this routine in which the user may be interested are stored in the `W` array, an argument of subroutine SDESOL.

- YMAX - array of maximum magnitudes of the independent variables, y_i , up to the current time (or one, if less than one). This is stored beginning at location $7*NY + NL + 1$ of the `W` array.
- ER - This is the array of differences between the predicted and corrected values of the variables, y_i , and is proportional to the estimated error. This array is stored beginning in location $8*NY + NL + 1$ of the `W` array.

This subroutine incorporates a restart capability. In order to restart from a previous point without beginning again with a first order method, it is necessary to have saved a number of variables internal to LDASUB, and then restore them before calling SDESOL again. To save the internal parameters, the user calls subroutine LDASAV(SAV). Here SAV is an array of length 29 in which the values to be saved will be stored. In addition to SAV, the user must also save a number of arrays in the calling sequence of LDASUB, and this is most easily accomplished by saving the `W` array in the calling sequence for SDESOL. Once these arrays have been saved, along with the other simple parameters in the calling sequence (`Y` and `YL` need not be saved), the user is free to

use the package to solve a different problem, or to terminate the computer run, to be restarted later.

At the time the problem is to be restarted, the user calls subroutine LDARST(SAV), where SAV is the array of values obtained previously by calling LDASAV. This restores internal values in LDASUB. The user then calls SDESOL with the same simple parameters and the W array as before, except that JSKF = - 1 and a new end time, TEND, is provided. Restoration of values (including Y and YL) in LDASUB is completed and solution of the problem resumes.

If the user desires to change the error tolerance, number of variables in the error test, or maximum order to be used, the user must make a new initial call to SDESOL, that is, set JSKF = 0 .

4.1.3 Subroutine COPYZ

This subroutine simply transfers the contents of one array into another array.

4.2 Supporting Subroutines

This group of subroutines must, at least in part, be supplied by the user. The user must supply at least one subroutine, DIFFUN. For better efficiency, the user should supply a subroutine, JACMAT, to explicitly evaluate the Jacobian, although a version which approximates the Jacobian by numerical differencing is given in the appendix. To take advantage of sparsity or other features of his problem, the user will need to supply the subroutine NUTSL to solve the systems of equations (2.4). For certain problems the user may have to supply

subroutine DERVAL to calculate the initial values of the derivatives.

We discuss the requirements of these subroutines in turn.

4.2.1 Subroutine DIFFUN

This subroutine simply evaluates the equations (2.1) at a given time and values of y , \dot{y} , and V . Other parameters in the function definition must be transmitted from the calling program via COMMON or some other device, determined by the user.

The calling sequence is

CALL DIFFUN (Y, YL, T, HINV, DY), where the parameters are defined as follows.

- Y - input. Same as in SDESOL. This array contains the current values of the variables y_i and their (scaled) derivatives.
- YL - input. Same as in SDESOL. This array contains the current values of the linear variables.
- T - input. Current time.
- HINV - input. $1/h$, where h is the current stepsize.
- DY - output. Array of function values.

4.2.2 Subroutine JACMAT

This subroutine evaluates the Jacobian matrix J , equation (2.5) at the given time and current values of the dependent variables, order, and stepsize. A version of JACMAT which approximates J by numerical differencing is given in the appendix. For maximum efficiency, the user should supply the explicit representation of the Jacobian. Because the Jacobian is used to solve for the quasi-Newton iterates, it is not

necessary for the Jacobian to be exact. Thus the user should consider the possibility of approximations which reduce the total number of computations in this step, with due regard for the fact that a smaller timestep may be required to obtain convergence of the corrector within three iterations.

The calling sequence for this subroutine is

CALL JACMAT (Y, YL, T, HINV, A2, N, NY, EPS, DY, F1, PW), where the parameters are defined as follows.

- Y - input. Same as in SDESOL, Y contains the current values of the variables y_i and their (scaled) derivatives.
- YL - input. Same as in SDESOL. This array contains the current values of the linear variables.
- T - input. Current time.
- HINV - input. $1/h$, where h is the current stepsize.
- A2 - input. The constant α_0/β_0 from LDASUB.
- N - input. Total number of variables.
- NY - input. Number of differential and nonlinear variables.
- EPS - input. Error constant from LDASUB, $\sqrt{M} \cdot \text{RMSEPS}$.
- DY - input. Array of current function values.
- F1 - scratch array of N locations available for use by this routine.
- PW - output. The Jacobian matrix J, or an approximation, calculated in JACMAT and returned to calling program. This matrix is used in subroutine NUITSL and the storage mode must agree between the two subroutines.

4.2.3 Subroutine NUITSL

This subroutine solves the equations (2.4) for the quasi-Newton iterates. This subroutine will normally be supplied by the user, although versions which solve the system by elimination methods and iterative methods, respectively, are given in the examples in the appendix. This subroutine will often be modified or replaced by the user to take advantage of sparsity or other features of his problem in connection with JACMAT, of course.

The calling sequence for this subroutine is
CALL NUITSL (PW, DY, F1, N, NY, EPS, YMAX, NEWPW, KRET), where the parameters are defined as follows.

- PW - input. The Jacobian matrix J computed in JACMAT.
- DY - input. Right hand side of the linear system to be solved.
- F1 - output. The solution is returned in the array F1 .
- N - input. Total number of variables.
- NY - input. Number of differential and nonlinear variables.
- EPS - input. Error constant from LDASUB, \sqrt{M} · RMSEPS .
- YMAX - input. Array of maximum magnitudes of y_i up to the current time (or one if maximum magnitude is less than one).
- NEWPW - input. Indicates whether a new J matrix has been computed since the last entry to NUITSL.
 - = 1, indicates this is a new J matrix. If any preprocessing, such as LU decomposition is to be done, the preprocessing should be done and NEWPW set to zero.
 - = 0, indicates the J matrix is the same as on the previous entry to NUITSL.

KRET - output. Return indicator r .

= 0 , normal return

= 1 , error return, solution of equations not obtained.

Note that the parameters EPS and YMAX are useful if an iterative method is used for solution of the equations. Because the solution represents corrections to the predicted value, and corrections to that, the solution is small compared to the dependent variable values. Hence, compared to the YMAX array, the error tolerance can be fairly large. The following convergence criteria have been used, with great success.

Let δu_i denote the i^{th} component of the difference between successive iterates, with u_i being the i^{th} component of the current iterate.

Then the iteration is considered to have converged whenever

$$(i) \quad \sum_{i=1}^{NY} \left(\frac{\delta u_i}{Y_{\max_i}} \right)^2 < \left(\epsilon/100 \right)^2, \text{ or}$$

$$(ii) \quad \sum_{i=1}^{NY} \left(\frac{\delta u_i}{\max(|u_i|, \epsilon)} \right)^2 < \epsilon^2.$$

Condition (i) requires convergence to 2 digits more accuracy than the user has asked for in the solution of the system (2.1), relative to YMAX. Condition (ii) requires the same relative accuracy in u_i as is asked for by the user in the solution of the system (2.1), unless the solution is smaller than ϵ , in which case the change is compared to ϵ rather than $|u_i|$. This avoids difficulty if u_i is close to zero. The ϵ above is $\text{EPS} = \sqrt{M} \cdot \text{RMSEPS}$, where M and RMSEPS are inputs to SDESOL. Two versions of NUITSL incorporating iteration and this convergence test are given in the appendix.

4.2.4 Subroutine DERVAL

This subroutine solves for, or otherwise supplies the initial values of the derivatives, and possibly other variables. In some instances it may need to be supplied by the user. The standard version of DERVAL given in the appendix uses Newton's method to solve the first $m (=NY)$ of the equations (2.1) for $\dot{y}(t_0)$, assuming values for $y(t_0)$ and $V(t_0)$ have been supplied. To accomplish this, the matrix $\frac{\partial F}{\partial \dot{y}}$ is needed, and this is obtained by calling JACMAT with $h = 16^{-20}$, $A2 = -1$, and $N = NY$, implying $NL = 0$ for this call. Special care must be taken if in fact NL is not zero to assure that the matrix is computed and stored properly. The matrix returned is then $16^{20} \frac{\partial F}{\partial \dot{y}}$. A call to DIFFUN yields the function values $F(y(t_0), \dot{y}(t_0), t_0) + PV(t_0)$ where $\dot{y}(t_0)$ is the current iterate. Multiplication of the function values by 16^{20} and a call to NUITSL (again with $N = NY$) gives the Newton iterate. Of course, the same sort of special care as necessary in JACMAT is necessary in NUITSL.

Obviously the above scheme cannot work if $\frac{\partial F}{\partial \dot{y}}$ is singular, such as it would be if one of the equations is algebraic. In this instance the user must either devise his own version of DERVAL, or supply the values along with a dummy version of DERVAL. In an extreme case the user may simply set initial derivatives to zero. This will provide a poor predicted value on the first step, and will force an artificially small timestep for the first two steps. However, the overall penalty is generally small, as appropriate (corrected) values are computed at the first step, and after two steps the program quickly increases the timestep.

The calling sequence for this subroutine is

CALL DERVAL (Y, YL, T, N, NY, DY, KERET) , where the parameters are defined as follows.

- Y - input and output. Same as in SDESOL. On entry Y(1,i) contains the initial values of the variables y_i . On return, the values of the derivatives are stored in Y(2 i) .
- YL - input. Same as in SDESOL. This array contains the initial values of the linear variables.
- T - input. initial time
- N - input. Total number of variables.
- NY - input. Number of differential and nonlinear variables.
- W - The scratch array from SDESOL, can be used in any way needed by this subroutine.
- KERET - output. Return indicator
 - = 0 normal return
 - = 1 error return, initial values were not obtained.

5.0 Acknowledgement

The author wishes to express his thanks to Professors David Salinas and Dong Nguyen of the Mechanical Engineering Department at the Naval Postgraduate School. They supplied the initial applications and encouragement for this work. They have continued to support it through valuable discussions with the author throughout the development period.

Appendix 1: Program Listings

The following are listings of the basic subroutine package and supporting subroutines which are of general use. For simple problems the user only needs to supply a calling program and a subroutine, DIFFUN, to evaluate the equations. Use of the NUITSL routine in computer facilities which do not subscribe to the IMSL package will necessitate modifications to replace LUDATF with another LU decomposition routine, and LUELMF with another forward and backward substitution routine.

```
SLBRoutine SDESOL (Y,YL,T,TEND,NY,NL,M,JSKF,MAXDER,IPRT,H,HMIN,
1HMAX,RMSEPS,W)
```

SLBRoutine SDESOL IS A DRIVER ROUTINE FOR SUBROUTINE LDASUB.
ITS PURPOSE IS TO SET UP THE NECESSARY REFERENCES TO A LARGE
BLOCK OF AUXILIARY STORAGE, AND OBTAIN INITIAL VALUES OF
DERIVATIVES.
THE CALLING SEQUENCE FOR SDESOL IS

```
CALL SDESOL(Y,YL,T,TEND,NY,NL,M,JSKF,MAXDER,IPRT,H,HMIN,HMAX,RMSEPS,W)
```

WHERE THE PARAMETERS ARE DEFINED AS FOLLOWS.

Y - ARRAY DIMENSIONED (7,NY). THIS ARRAY CONTAINS THE
DEPENDENT VARIABLES AND THEIR SCALED DERIVATIVES.
Y(J+1,I) CONTAINS THE J-TH DERIVATIVE OF THE I-TH VAR-
IABLE TIMES H**J/J-FACTORIAL, WHERE H IS THE CURRENT
STEP SIZE. ON FIRST ENTRY THE CALLER SUPPLIES THE
INITIAL VALUES OF EACH VARIABLE IN Y(1,I). ON SUB-
SEQUENT ENTRIES IT IS ASSUMED THE ARRAY HAS NOT
BEEN CHANGED. TO INTERPOLATE TO NON-MESH POINTS,
THESE VALUES CAN BE USED AS FOLLOWS. IF H IS THE
CURRENT STEP SIZE AND VALUES AT TIME T+E ARE
NEEDED, LET S = E/H AND THEN

JS
I-TH VARIABLE AT T+E IS SUM Y(J+1,I)*S**J
J=0

THE VALUE OF JS IS OBTAINED IN THE CALLING PROGRAM
BY JS = IABS(JSKF/10)

YL - ARRAY OF NL VARIABLES WHICH APPEAR LINEARLY.
T - CURRENT VALUE OF THE INDEPENDENT VARIABLE (TIME)
TEND - END TIME
NY - NUMBER OF DIFFERENTIAL EQUATIONS AND NONLINEAR
VARIABLES.
NL - NUMBER OF LINEAR VARIABLES
M - NUMBER OF VARIABLES INCLUDED IN THE ERROR TEST
JSKF - AN INDICATOR USED BOTH ON INPUT AND OUTPUT
ON INPUT, JSKF = -1 INDICATES A RESTART CALL TO
SDESOL. JSKF = 0 INDICATES AN INITIAL CALL TO
SDESOL. JSKF > 0 INDICATES A CONTINUATION OF THE
PREVIOUS CALL TO SDESOL. JSKF < -1 MAY HAVE RESULTED
FROM THE USER NEGLECTING TO TEST FOR ERROR RETURNS
FROM SDESOL. BECAUSE OF THIS POSSIBILITY, JSKF < -1
RESULTS IN TERMINATION OF THE RUN WITH THE
APPROPRIATE COMMENT.
ON OUTPUT, JSKF CONSISTS OF TWO DIGITS AND SIGN,
+ OR - QP. Q IS THE ORDER OF THE FORMULA CURRENTLY
BEING USED. P INDICATES THE TYPE OF RETURN, AS
FOLLOWS.
JSKF > 0, P = 1 IS THE NORMAL RETURN
JSKF < 0 IS AN ERROR RETURN, WITH THE FOLLOWING
MEANINGS.
P = 1 ERROR TEST FAILURE FOR H > HMIN
P = 3 CORRECTOR FAILED TO CONVERGE FOR H > HMIN
P = 4 CORRECTOR FAILED TO CONVERGE FOR FIRST
ORDER METHOD
P = 5 ERROR RETURN FROM SUBROUTINE NUTSL
P = 6 ERROR RETURN FROM SUBROUTINE DERVAL
MAXDER - MAXIMUM ORDER DERIVATIVE THAT SHOULD BE USED IN
METHOD. IT MUST BE NO GREATER THAN SIX.
IPRT - INTERNAL PRINT CONTROL INDICATOR FOR LDASUB.
IPRT = 0 NO PRINT
IPRT > 0 PRINT COUNTERS, STEP SIZE, CURRENT TIME
AND VALUES OF DEPENDENT VARIABLES AT
EACH STEP.
H - CURRENT STEP SIZE. AN INITIAL VALUE MUST BE SUPPLIED
BUT NEED NOT BE THE ONE WHICH MUST BE USED, SINCE THE
SUBROUTINE WILL CHOOSE A SMALLER ONE IF NECESSARY TO
KEEP THE ERROR PER STEP SMALLER THAN THE SPECIFIED
VALUE. IT IS BETTER TO UNDERESTIMATE THE INITIAL
STEP SIZE THAN TO OVERESTIMATE IT. THE STEP SIZE IS


```

C          NORMALLY NOT CHANGED BY THE USER.
C          HMIN - MINIMUM STEPSIZE ALLOWED
C          HMAX - MAXIMUM STEPSIZE ALLOWED
C          RMSEPS - THE ERROR TEST CONSTANT. THE ROOT-MEAN-SQUARE OF
C                  THE SINGLE STEP ERROR ESTIMATES,  $FR(I)$ , DIVIDED BY
C                   $YMAX(I) = (\text{MAXIMUM TO CURRENT TIME OF } Y(I))$  MUST BE
C                  LESS THAN EPS. THE STEPSIZE AND/OR THE ORDER
C                  ARE VARIED TO ACHIEVE THIS.
C          W - SCRATCH STORAGE ARRAY. MUST BE AT LEAST  $13*NY + 5*NL$ 
C              LOCATIONS, PLUS THOSE REQUIRED FOR STORAGE OF THE
C              MATRIX PW (SEE DESCRIPTION OF SUBROUTINE JACMAT).
C              THE STORAGE OF PW WILL NORMALLY REQUIRE NO MORE THAN
C               $N**2 + 2*N$  LOCATIONS, AND IF COMPACT STORAGE TECH-
C              NIQUES ARE USED, CAN BE MUCH FEWER.
C
C-----
C          DIMENSION Y(7,1), YL(1), W(1)
C          IF (JSKF.GT.0) GO TO 120
C          IF (JSKF.LT.-1) GO TO 140
C          N = NY+NL
C          IF (JSKF.LT.0) GO TO 110
C
C          IF THIS IS THE FIRST ENTRY, OBTAIN VALUES OF THE DERIVATIVES.
C          CALL Derval (Y, YL, T, N, NY, W, KRETR)
C          IF (KRETF.NE.0) GO TO 130
C
C          NOW SET UP STORAGE BLOCKS IN THE W ARRAY. THIS NEEDS TO BE DONE
C          ONLY INITIALLY AND ON RESTARTS.
C
C          THE ARRAY SAVE STARTS AT LOCATION 1 IN THE W ARRAY
C          THE ARRAY YLSV STARTS AT LOCATION NSVL IN THE W ARRAY
C          THE ARRAY YMAX STARTS AT LOCATION NYMAX IN THE W ARRAY
C          THE ARRAY ER STARTS AT LOCATION NER IN THE W ARRAY
C          THE ARRAY ESV STARTS AT LOCATION NESV IN THE W ARRAY
C          THE ARRAY F1 STARTS AT LOCATION NF1 IN THE W ARRAY
C          THE ARRAY DY STARTS AT LOCATION NCY IN THE W ARRAY
C          THE MATRIX PW STARTS AT LOCATION NPW IN THE W ARRAY
C
C          110 NSVL = 7*NY+1
C              NYMAX = NSVL+NL
C              NER = NYMAX+NY
C              NESV = NER+NY
C              NF1 = NESV+NY
C              NCY = NF1+N
C              NPW = NCY+N
C          120 JS = JSKF
C              CALL LDASUB (Y, YL, T, TEND, N, NY, M, JS, KF, MAXDER, IPRT, F, HMIN, HMAX,
C              1 RMSEPS, W, W(NSVL), W(NYMAX), W(NER), W(NESV), W(NF1), W(NCY), W(NPW))
C
C          CODE JSKF ON RETURN FROM LDASUB
C
C          JSKF = ISIGN(JS*10+IABS(KF), KF)
C          RETURN
C          130 JSKF = -6
C              RETURN
C          140 PRINT 1, JSKF
C              STOP
C
C          1 FORMAT ('OIT IS AN ERROR TO ENTER SDESOL WITH JSKF = ', I10//
C          1 ' ' RUN HAS BEEN TERMINATED.')
C          END
C
C          SUBROUTINE LDASUB (Y, YL, T, TEND, N, NY, M, JSTART, KFLAG, MAXDR, IPRT, H,
C          1 HMIN, HMAX, RMSEPS, SAVE, YLSV, YMAX, ER, ESV, F1, DY, PW)
C
C          SUBROUTINE LDASUB IS A MODIFICATION OF SUBROUTINE DFASUB
C          WHICH IS DUE TO R. L. BROWN AND C. W. GEAR. DFASUB IS DOCUMENTED
C          IN THE REPORT
C          DOCUMENTATION FOR DFASUB--
C          BY R. L. BROWN AND C. W. GEAR
C          REPORT UIUCDCS-R-73-575, JULY 1973
C          UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
C          URBANA, ILLINOIS 61801

```

THIS REPORT IS AVAILABLE FROM THE NATIONAL TECHNICAL INFORMATION SERVICE OF THE U. S. DEPARTMENT OF COMMERCE UNDER ACCESSION NUMBER CCO-1469-225.

THE MODIFICATION HERE IS DOCUMENTED IN THE REPORT
A PROGRAM FOR THE NUMERICAL SOLUTION OF LARGE SPARSE SYSTEMS OF
ALGEBRAIC AND IMPLICITLY DEFINED STIFF DIFFERENTIAL EQUATIONS
BY RICHARD FRANK
REPORT NPS53F-76051, MAY 1976
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93940

THE CALLING SEQUENCE FOR LDASUB IS

CALL LDASUB(Y, YL, T, TEND, W, NY, M, JSTART, KFLAG, MAXOR, IPRT, H, HMIN,
HMAX, RMSEPS, SAVE, YLSV, YMAX, ER, ESV, F1, DY, PW)

WHERE THE PARAMETERS ARE DEFINED AS FOLLOWS.

Y - ARRAY DIMENSIONED (7, NY). THIS ARRAY CONTAINS THE
DEPENDENT VARIABLES AND THEIR SCALED DERIVATIVES.
Y(J+1, I) CONTAINS THE J-TH DERIVATIVE OF THE I-TH VAR-
IABLE TIMES H**J/J-FACTORIAL, WHERE H IS THE CURRENT
STEP SIZE. ON FIRST ENTRY THE CALLER SUPPLIES THE
INITIAL VALUES OF EACH VARIABLE IN Y(1, I) AND AN
ESTIMATE OF THE INITIAL VALUES OF THE DERIVATIVES
IN Y(2, I). ON SUBSEQUENT ENTRIES IT IS ASSUMED THAT
THE ARRAY HAS NOT BEEN CHANGED. TO INTERPOLATE TO
NON-MESH POINTS, THESE VALUES CAN BE USED AS FOLLOWS.
IF H IS THE CURRENT STEP SIZE AND VALUES AT TIME T+E
NEEDED, LET S = E/H AND THEN

I-TH VARIABLE AT T+E IS $\sum_{J=0}^{NQ} Y(J+1, I) * S^J$

THE VALUE OF NQ IS OBTAINED IN THE CALLING PROGRAM
BY NQ = JSTART.

YL - ARRAY OF NL = N - NY VARIABLES WHICH APPEAR LINEARLY.
T - THE USER SUPPLIES INITIAL VALUES FOR THESE VARIABLES.
TEND - CURRENT VALUE OF THE INDEPENDENT VARIABLE (TIME)
N - END TIME
NY - TOTAL NUMBER OF VARIABLES
M - NUMBER OF DIFFERENTIAL EQUATIONS AND NONLINEAR
VARIABLES.
JSTART - NUMBER OF VARIABLES INCLUDED IN THE ERROR TEST.
ON INPUT JSTART HAS THE FOLLOWING MEANINGS.
ON OUTPUT JSTART IS SET TO THE VALUE OF NQ, THE
ORDER OF THE FORMULA CURRENTLY BEING USED.

<0 THIS INDICATES A RE-START FROM A PREVIOUS
POINT FOLLOWING TERMINATION OF THE RUN OR
SOLUTION OF ANOTHER PROBLEM DURING THE SAME
RUN. PARAMETERS IN THE CALLING SEQUENCE
MUST HAVE BEEN PRESERVED FROM THE PREVIOUS
USE, PARTICULARLY THE ARRAYS
SAVE, YLSV, ESV, AND PW.
THESE ARRAYS MUST BE SAVED AFTER A CALL
TO SUBROUTINE LDASAV, WHICH ALSO SAVES
NECESSARY PARAMETERS INTERNAL TO LDASUB.

=0 INDICATES AN INITIAL CALL TO LDASUB. THE
ROUTINE INITIALIZES ITSELF, SCALES THE
DERIVATIVES IN Y(2, I) AND THEN PERFORMS THE
INTEGRATION UNTIL T > TEND.

>0 INDICATES THE SOLUTION IS TO BE CONTINUED.
AFTER THE INITIAL ENTRY IT IS NEITHER
DESIRABLE NOR NECESSARY TO RE-ENTER WITH
JSTART = 0, SINCE THIS RE-INITIALIZES
THE CODE, BEGINNING WITH A FIRST ORDER
METHOD AGAIN.

```

C      KFLAG - THE COMPLETION CODE INDICATOR, WITH THE FOLLOWING MEANINGS
C      +1 THE INTEGRATION WAS SUCCESSFUL
C      -1 ERROR TEST FAILURE FOR  $H > H_{MIN}$ 
C      -3 CORRECTOR FAILED TO CONVERGE FOR  $H > H_{MIN}$ 
C      -4 CORRECTOR FAILED TO CONVERGE FOR FIRST ORDER METHOD
C      -5 ERROR RETURN FROM SUBROUTINE NUTSL
C      MAXOR - MAXIMUM ORDER DERIVATIVE THAT SHOULD BE USED IN THE METHOD. IT MUST BE NO GREATER THAN SIX. IF IT IS GREATER THAN SIX, THE MAXIMUM ORDER USED WILL BE SIX.
C      IPRT - INTERNAL PRINT CONTROL INDICATOR
C      = 0 NO PRINT
C      > 0 PRINT COUNTERS, STEPSIZE, CURRENT TIME AND VALUES OF DEPENDENT VARIABLES AT EACH STEP.
C      H - CURRENT STEPSIZE. AN INITIAL VALUE MUST BE SUPPLIED BUT NEED NOT BE THE ONE WHICH WILL BE USED, SINCE THE SUBROUTINE WILL CHOOSE A SMALLER ONE IF NECESSARY TO KEEP THE ERROR PER STEP SMALLER THAN THE SPECIFIED VALUE. IT IS BETTER TO UNDERESTIMATE THE INITIAL STEPSIZE THAN TO OVERESTIMATE IT. THE STEPSIZE IS NORMALLY NOT CHANGED BY THE USER.
C      HMIN - MINIMUM STEPSIZE ALLOWED
C      HMAX - MAXIMUM STEPSIZE ALLOWED
C      RMSEPS - THE ERROR TEST CONSTANT. THE ROOT-MEAN-SQUARE OF THE SINGLE STEP ERROR ESTIMATES,  $ER(I)$ , DIVIDED BY  $YMAX(I) = (\text{MAXIMUM TO CURRENT TIME OF } Y(I))$  MUST BE LESS THAN RMSEPS. THE STEPSIZE AND/OR ORDER ARE VARIED TO ACHIEVE THIS.
C      SAVE - AN ARRAY OF LENGTH AT LEAST  $7*NY$ 
C      YLSV - AN ARRAY OF LENGTH AT LEAST  $NL$ 
C      YMAX - A VECTOR OF LENGTH  $NY$  WHICH CONTAINS THE MAXIMUM OF EACH  $Y$  SEEN SO FAR. ON THE FIRST CALL, THESE WILL BE INITIALIZED AS  $YMAX(I) = \max(1, |Y(1, I)|)$ 
C      ER - A VECTOR OF LENGTH  $NY$ 
C      ESV - A VECTOR OF LENGTH  $NY$ 
C      F1 - A VECTOR OF LENGTH  $N = NY + NL$ 
C      DY - A VECTOR OF LENGTH  $N = NY + NL$ 
C      PW - AN ARRAY IN WHICH THE  $J$  MATRIX COMPUTED IN SUBROUTINE JACMAT WILL BE STORED. SIZE WHICH MUST BE ALLOWED IS DETERMINED BY THE STORAGE TECHNIQUE USED FOR IT, BUT NORMALLY WON'T BE MORE THAN  $N**2 + 2*N$  LOCATIONS, THE LATTER  $2*N$  BEING REQUIRED BY THE LINEAR EQUATION SOLVER.
C
C-----
C      DIMENSION Y(7,1), YL(1), SAVE(7,1), YMAX(1), ER(1), YLSV(1), F1(1)
C      1. PERT(6,3), COF(21), ESV(1), DY(1), PW(1), SAV(1), A(29)
C      EQUIVALENCE (A(8),BND), (A(9),BR), (A(10),F), (A(11),EDW),
C      1(A(12),ENQ1), (A(13),FNQ2), (A(14),ENQ3), (A(15),EPS), (A(16),EUP)
C      2,(A(17),FNW), (A(18),PEPSH), (A(19),IDUP), (A(20),IWEAL),
C      3(A(21),K), (A(22),LCOPYL), (A(23),LCOPYY), (A(24),MAXDER),
C      4(A(25),M1), (A(26),NL), (A(27),NQ), (A(28),NS), (A(29),NW)
C-----
C
C      THE COEFFICIENTS IN THE PERT ARRAY ARE USED FOR ERROR TESTING AND CHANGING STEPSIZE AND NEED TO BE ACCURATE TO ONLY A FEW DIGITS.
C-----
C      DATA PERT/4.,9.,16.,25.,36.,49.,9.,16.,25.,36.,49.,64.,1.,1.,.25,
C      12.7889E-2,1.70569E-3,6.83929E-5/
C-----
C
C      THE ENTRIES IN THE COF ARRAY ARE THE COEFFICIENTS FOR THE STIFFLY STABLE METHODS USED IN THIS PROGRAM AND ARE TO BE THE MACHINE PRECISION EQUIVALENTS OF THE FOLLOWING CONSTANTS.
C-----
C      -1
C      -3/2, -1/2
C      -11/6, -1, -1/6
C      -25/12, -35/24, -5/12, -1/24
C      -137/60, -15/8, -17/24, -1/8, -1/120
C      -147/60, -203/90, -49/48, -35/144, -7/240, -1/720

```

C			LDA	1620
C			LDA	1630
		DATA COF/-1.,-1.5,-.5,-1.833333,-1.,-.1666667,-2.083333,-1.458333,	LCA	1640
		1-.4166667,-.04166667,-2.283333,-1.875,-.7083333,-.125,-.008333333,	LCA	1650
		2-2.45,-2.255556,-1.020833,-.2430556,-.02916667,-.001388889/	LDA	1660
		IF (JSTART) 100,110,150	LDA	1670
C			LDA	1680
C		IF THIS IS A RESTART ENTRY, RESTORE Y AND YL FROM THE SAVE AND	LCA	1690
C		YLSV ARRAYS, WHERE THEY WERE SAVED BY A PREVIOUS CALL TO LDASAV.	LCA	1700
C			LDA	1710
100		CALL COPYZ (Y,SAVE,LCOPYY)	LCA	1720
		CALL COPYZ (YL,YLSV,LCOPYL)	LCA	1730
		GC TO 150	LDA	1740
C			LDA	1750
C		IF THIS IS THE FIRST CALL, INITIALIZE YMAX, SCALE DERIVATIVES, AND	LDA	1760
C		INITIALIZE INDICATORS AND SET ORDER TO ONE.	LDA	1770
C		FOR DOUBLE PRECISION, SET LCOPYL = 14*NY AND LCOPYL = 2*NL IF	LDA	1780
C		SLBROUTINE COPYZ IS IN SINGLE PRECISION.	LCA	1790
C			LDA	1800
110		NL = N-NY	LDA	1810
		LCOPYY = 7*NY	LCA	1820
		LCOPYL = NL	LCA	1830
		M1 = MIN0(N,NY)	LDA	1840
		EPS = SCRT(FLOAT(M1))*RMSEPS	LDA	1850
		MAXCER = MIN0(MAXCR,6)	LCA	1860
		IF (IPRT.LE.0) GO TO 120	LDA	1870
		PRINT 3, N,NL,RMSEPS,TEND,H	LCA	1880
		PRINT 4	LDA	1890
120		NS = 0	LDA	1900
		NW = 0	LDA	1910
C			LDA	1920
		GO 130 J=1,NY	LDA	1930
		YMAX(J) = YMAX1(1.,ABS(Y(1,J)))	LDA	1940
130		Y(2,J) = Y(2,J)*H	LDA	1950
C			LDA	1960
		NC = 1	LDA	1970
		BR = 1.	LDA	1980
		ASSIGN 190 TO IRET	LCA	1990
C			LDA	2000
C		SET COEFFICIENTS FOR THE ORDER CURRENTLY BEING USED.	LCA	2010
C		E IS A TEST FOR ERRORS OF THE CURRENT ORDER NQ	LCA	2020
C		EDWN IS TO TEST FOR INCREASING THE ORDER, EDWN FOR DECREASING THE	LCA	2030
C		ORDER.	LCA	2040
C			LDA	2050
140		K = NC*(NQ-1)/2	LDA	2060
		CALL COPYZ (A(2),COF(K+1),NQ)	LDA	2070
		K = NC+1	LCA	2080
		EDOWN = NC	LCA	2090
		ENC1 = .5/NC	LCA	2100
		ENC2 = .5/K	LCA	2110
		ENC3 = .5/(NC+2)	LCA	2120
		PEPSH = EPS**2	LDA	2130
		E = PERT(ENC,1)*PEPSH	LCA	2140
		EDWN = PERT(ENC,2)*PEPSH	LCA	2150
		EDWN = PERT(ENC,3)*PEPSH	LDA	2160
		BNC = (EPS*ENC3)**2	LDA	2170
		IWEVAL = 1	LDA	2180
		GC TO IRET, (190,200,490,570)	LCA	2190
150		IF (H.EQ.HNEW) GO TO 190	LCA	2200
C			LDA	2210
C		IF CALLER HAS CHANGED H, RESCALE DERIVATIVES TO REFLECT THAT HNEW	LCA	2220
C		WAS USED ON THE LAST CALL.	LCA	2230
C			LDA	2240
		R = H/HNEW	LCA	2250
		ASSIGN 190 TO IRET	LCA	2260
		GC TO 610	LCA	2270
C			LDA	2280
C		SET JSTART TO NQ, THE CURRENT ORDER OF THE METHOD, BEFORE EXIT,	LDA	2290
C		AND SAVE THE CURRENT STEPSIZE IN HNEW.	LDA	2300
C			LDA	2310
160		JSTART = NQ	LDA	2320
		HNEW = H	LDA	2330
		RETURN	LCA	2340
170		NS = NS+1	LCA	2350
		IF (IPRT.LE.0) GO TO 180	LCA	2360

C	PRINT DATA IF DESIRED BY USER	LOA	2370
C	PRINT 1, NS, NW, NQ, H, T, (Y(1,I), I=1, NY)	LOA	2380
	IF (NL.GT.0) PRINT 2, (YL(1), I=1, NL)	LOA	2390
180	CONTINUE	LOA	2400
	IF (KFLAG.LT.0) GO TO 160	LOA	2410
	IF (T.GE.TEND) GO TO 160	LOA	2420
C	TAKE ANOTHER STEP IF T < TEND	LOA	2430
C	JSTART = 1	LOA	2440
C	SAVE DATA FOR TRIAL WITH A SMALLER TIMESTEP IF THIS STEP FAILS	LOA	2450
C	CALL COPYZ (SAVE, Y, LCOPIY)	LOA	2460
190	CALL COPYZ (YLSV, YL, LCOPLY)	LOA	2470
	RACUM = 1.	LOA	2480
	KFLAG = 1	LOA	2490
	HOLD = H	LOA	2500
	NCOLD = NQ	LOA	2510
	TOLD = T	LOA	2520
200	T = T+H	LOA	2530
	HINV = 1./H	LOA	2540
C	COMPUTE PREDICTED VALUES BY EFFECTIVELY MULTIPLYING DERIVATIVE	LOA	2550
C	VECTOR BY PASCAL TRIANGLE MATRIX	LOA	2560
C	DO 210 J=2,K	LOA	2570
	J3 = K+J-1	LOA	2580
C	DO 210 J1=J,K	LOA	2590
	J2 = J3-J1	LOA	2600
C	DO 210 I=1, NY	LOA	2610
210	Y4J2,I) = Y(J2,I)+Y(J2+1,I)	LOA	2620
C	DO 220 I=1, NY	LOA	2630
220	ER(I) = C.	LOA	2640
C	DO UP TO THREE CORRECTOR ITERATIONS. CONVERGENCE IS OBTAINED WHEN	LOA	2650
C	CHANGES ARE LESS THAN BND WHICH IS DEPENDENT ON THE ERROR TEST	LOA	2660
C	CONSTANT. THE SUM OF CORRECTIONS IS ACCUMULATED IN ER(I). IT IS	LOA	2670
C	EQUAL TO THE K-TH DERIVATIVE OF Y TIMES H**K/(K-FACTORIAL*A(K)),	LOA	2680
C	AND THIS IS PROPORTIONAL TO THE ACTUAL ERROR TO THE LOWEST POWER	LOA	2690
C	OF H PRESENT, WHICH IS H**K.	LOA	2700
C	DO 270 L=1,3	LOA	2710
	CALL DIFFUN (Y, YL, T, HINV, DY)	LOA	2720
	IF (IWEVAL.LT.1) GO TO 230	LOA	2730
C	IF THERE HAS BEEN A CHANGE OF ORDER OR THERE HAS BEEN TROUBLE	LOA	2740
C	WITH CONVERGENCE, PW IS RE-EVALUATED PRIOR TO STARTING THE	LOA	2750
C	CORRECTOR ITERATION. IWEVAL IS THEN SET TO -1 AS AN INDICATOR	LOA	2760
C	THAT IT HAS BEEN DONE. NEWPW IS SET NONZERO TO INDICATE TO	LOA	2770
C	SUBROUTINE NUTSL THAT A NEW PW HAS BEEN PROVIDED.	LOA	2780
C	CALL JACMAT (Y, YL, T, HINV, A(2), N, NY, EPS, DY, F1, PW)	LOA	2790
	KFLAG = 1	LOA	2800
	IWEVAL = -1	LOA	2810
	Nk = NW+1	LOA	2820
	NEWPW = 1	LOA	2830
230	CALL NUTSL (PW, DY, F1, N, NY, EPS, YMAX, NEWPW, KRRET)	LOA	2840
	IF (KRRET.NE.0) GO TO 600	LOA	2850
	IF (NL.LE.0) GO TO 250	LOA	2860
C	DO 240 I=1, NL	LOA	2870
240	YL(I) = YL(I)-F1(I+NY)	LOA	2880
C	250 CONTINUE	LOA	2890
	DEL = 0.	LOA	2900

C	DO 260 I=1,NY	LDA	3120
	Y(1,I) = Y(1,I)-F1(I)	LDA	3130
	Y(2,I) = Y(2,I)+A(2)*F1(I)	LDA	3140
	ER(I) = ER(I)+F1(I)	LDA	3150
	DEL = DEL+(F1(I)/AMAX1(YMAX(I),ABS(Y(1,I))))**2	LDA	3160
260	CONTINUE	LDA	3170
C	IF (L.GE.2) ER = AMAX1(.9*BR,DEL/DEL1)	LDA	3180
	DEL1 = DEL	LDA	3190
	IF (AMIN1(DEL,BR*DEL*2.).LE.BND) GO TO 330	LDA	3200
270	CONTINUE	LDA	3210
C	-----	LDA	3220
C	THE CORRECTOR ITERATION FAILED TO CONVERGE IN 3 TRIES. VARIOUS	LDA	3230
C	POSSIBILITIES ARE CHECKED FOR. IF H IS ALREADY HMIN AND PW HAS	LDA	3240
C	ALREADY BEEN RE-EVALUATED, A NO CONVERGENCE EXIT IS TAKEN.	LDA	3250
C	OTHERWISE THE MATRIX PW IS RE-EVALUATED AND/OR (IN THAT ORDER) THE	LDA	3260
C	STEP IS REDUCED TO TRY AND GET CONVERGENCE.	LDA	3270
C	-----	LDA	3280
	T = TOLC	LDA	3290
	IF (IWEVAL) 280,300,290	LDA	3300
280	IF (H.LE.HMIN*1.00001) GO TO 310	LDA	3310
290	RACUM = RACUM*.25	LDA	3320
300	CONTINUE	LDA	3330
	GC TC 560	LDA	3340
310	KFLAG = -3	LDA	3350
C	-----	LDA	3360
C	RESTORE Y AND YL AFTER CONVERGENCE FAILURE	LDA	3370
C	-----	LDA	3380
320	CALL COPYZ (Y,SAVE,LCOPYY)	LDA	3390
	CALL COPYZ (YL,YLSV,LCOPYL)	LDA	3400
	H = HOLC	LDA	3410
	NC = NCCLD	LDA	3420
	GO TO 170	LDA	3430
C	-----	LDA	3440
C	THE CORRECTOR CONVERGED, SO NOW THE ERROR TEST IS MADE.	LDA	3450
C	-----	LDA	3460
330	C = 0.	LDA	3470
C	-----	LDA	3480
	DO 340 I=1,M1	LDA	3490
	YM = AMAX1(AES(Y(1,I)),YMAX(I))	LDA	3500
340	D = D+(ER(I)/YM)**2	LDA	3510
C	-----	LDA	3520
	IWEVAL = 0	LDA	3530
	IF (C.GT.E) GO TO 380	LDA	3540
C	-----	LDA	3550
C	THE ERROR TEST IS OKAY, SO THE STEP IS ACCEPTED. IF IDOUB	LDA	3560
C	NOW BECOMES NEGATIVE, A TEST IS MADE TO SEE IF THE STEP SIZE	LDA	3570
C	CAN BE INCREASED AT THIS ORDER OR ONE HIGHER OR ONE LOWER.	LDA	3580
C	THE CHANGE IS MADE ONLY IF THE STEP CAN BE INCREASED BY AT	LDA	3590
C	LEAST 10%. IDOUB IS SET TO NC TO PREVENT FURTHER TESTING	LDA	3600
C	FOR A WHILE. IF NO CHANGE IS MADE, IDOUB IS SET TO 9.	LDA	3610
C	-----	LDA	3620
	IF (K.LT.3) GO TO 360	LDA	3630
C	-----	LDA	3640
	DO 350 J=3,K	LDA	3650
C	-----	LDA	3660
	DO 350 I=1,NY	LDA	3670
350	Y(J,I) = Y(J,I)+A(J)*ER(I)	LDA	3680
C	-----	LDA	3690
360	KFLAG = 1	LDA	3700
	IDOUB = IDOUB-1	LDA	3710
	IF (IDOUB) 410,370,510	LDA	3720
370	CALL COPYZ (ESV,ER,M1)	LDA	3730
	GO TO 510	LDA	3740
C	-----	LDA	3750
C	THE ERROR TEST FAILED. IF JSTART = 0, THE DERIVATIVES IN THE	LDA	3760
C	SAVE ARRAY ARE UPDATED. TESTS ARE THEN MADE TO FIX THE STEPSIZE	LDA	3770
C	AND PERHAPS REDUCE THE ORDER. AFTER RESTORING AND SCALING THE	LDA	3780
C	Y VARIABLES, THE STEP IS RETRIED.	LDA	3790
C	-----	LDA	3800
380	IF (JSTART.GT.0) GO TO 400	LDA	3810
C	-----	LDA	3820
	DO 390 I=1,NY	LDA	3830
		LDA	3840
		LDA	3850
		LDA	3860

C	390	SAVE(2,I) = Y(2,I)	LDA	3870
C	400	KFLAG = KFLAG-2	LDA	3880
		IF (H.LE.HMIN) GO TO 550	LDA	3890
		T = TOLD	LDA	3900
		IF (KFLAG.LE.-5) GO TO 530	LDA	3910
C	410	PR2 = (C/E)**ENQ2*1.2	LDA	3920
		L = 0	LDA	3930
		IF (NQ.LE.1) GO TO 430	LDA	3940
		C = 0.	LDA	3950
C			LDA	3960
	420	J=1,M1	LDA	3970
		YM = AMAX1(ABS(Y(1,J)),YMAX(J))	LDA	3980
C	420	C = D+(Y(K,J)/YM)**2	LDA	3990
			LDA	4000
		PR1 = (C/EDWN)**ENQ1*1.3	LDA	4010
		IF (PR1.GE.PR2) GO TO 430	LDA	4020
		PR2 = PR1	LDA	4030
		L = -1	LDA	4040
C	430	IF (KFLAG.LT.0.OR.NQ.GE.MAXDER) GO TO 450	LDA	4050
		C = 0	LDA	4060
C			LDA	4070
	440	J=1,M1	LDA	4080
		YM = AMAX1(ABS(Y(1,J)),YMAX(J))	LDA	4090
C	440	C = D+((ER(J)-ESV(J))/YM)**2	LDA	4100
			LDA	4110
		PR1 = (C/EUF)**ENQ3*1.4	LDA	4120
		IF (PR1.GE.PR2) GO TO 450	LDA	4130
		PR2 = PR1	LDA	4140
		L = 1	LDA	4150
C	450	R = 1./AMAX1(PR2,1.E-5)	LDA	4160
		IF (KFLAG.LT.0.OR.R.GE.1.1) GO TO 460	LDA	4170
		ICDUB = 5	LDA	4180
		GO TO 510	LDA	4190
C	460	NEWQ = NC+L	LDA	4200
		K = NEWQ+1	LDA	4210
		IF (NEWQ.LE.NQ) GO TO 480	LDA	4220
		R1 = A(NEWQ)/FLOAT(NEWQ)	LDA	4230
C			LDA	4240
	470	J=1,NY	LDA	4250
		Y(K,J) = ER(J)*R1	LDA	4260
C			LDA	4270
	480	CONTINUE	LDA	4280
			LDA	4290
C		-----	LDA	4300
C		IF THE STEP WAS OKAY, SCALE THE Y VARIABLES IN ACCORDANCE	LDA	4310
C		WITH THE NEW VALUE OF H. IF KFLAG < 0, HOWEVER, USE THE	LDA	4320
C		SAVED VALUES (IN SAVE AND YLSV). IN EITHER CASE, IF THE ORDER	LDA	4330
C		HAS CHANGED IT IS NECESSARY TO FIX CERTAIN PARAMETERS BY CALLING	LDA	4340
C		THE PROGRAM SEGMENT AT STATEMENT NUMBER 140.	LDA	4350
C		-----	LDA	4360
		ICDUB = NQ	LDA	4370
		IF (NEWQ.EQ.NQ) GO TO 490	LDA	4380
		NC = NEWQ	LDA	4390
		ASSIGN 450 TO IRET	LDA	4400
		GO TO 140	LDA	4410
C	490	IF (KFLAG.GT.0) GO TO 500	LDA	4420
		RACUM = RACUM*R	LDA	4430
		GO TO 560	LDA	4440
C	500	R = AMAX1(AMIN1(HMAX/H,R),HMIN/H)	LDA	4450
		H = H*R	LDA	4460
		IHEVAL = 1	LDA	4470
		ASSIGN 510 TO IRET	LDA	4480
		GO TO 610	LDA	4490
C			LDA	4500
	510	CC 520 I=1,M1	LDA	4510
C	520	YMAX(I) = AMAX1(ABS(Y(1,I)),YMAX(I))	LDA	4520
			LDA	4530
		GO TO 170	LDA	4540
C		-----	LDA	4550
C		THE ERROR TEST HAS NOW FAILED THREE TIMES, SO THE DERIVATIVES ARE	LDA	4560
C		IN BAD SHAPE. RETURN TO FIRST ORDER METHOD AND TRY AGAIN. IF	LDA	4570
C		CCURSE, IF NC = 1 ALREADY, THEN THERE IS NO HOPE AND WE EXIT WITH	LDA	4580
C		KFLAG = -4.	LDA	4590
C		-----	LDA	4600
	530	IF (NQ.EQ.1) GO TO 540	LDA	4610

	NC = 1	LDA	462C
	ICCUB = 1	LDA	4630
	ASSIGN 570 TC IRET	LDA	4640
	GC TO 140	LDA	4650
540	NCOLD = 1	LDA	4660
	KFLAG = -4	LDA	4670
	GO TO 320	LDA	4680
550	KFLAG = -1	LDA	4690
	GC TO 170	LDA	4700
C	-----	LDA	4710
C	THIS SECTION RESTORES THE SAVED VALUES OF Y AND YL, SCALING THE	LDA	4720
C	Y DERIVATIVES AS NECESSARY, AND THEN RETURNS TO THE PREDICTOR LOOP.	LDA	4730
C	-----	LDA	4740
560	F = HCLD*RACUM	LDA	4750
	H = AMAX1(HMIN,AMIN1(H,HMAX))	LDA	4760
570	RACUM = F/HCLD	LDA	4770
	R1 = 1.	LDA	4780
C	-----	LDA	4790
	DO 580 J=2,K	LDA	4800
C	R1 = R1*RACUM	LDA	4810
C	-----	LDA	4820
	DO 580 I=1,NY	LDA	4830
580	Y(J,I) = SAVE(J,I)*R1	LDA	4840
C	-----	LDA	4850
	DO 590 I=1,NY	LDA	4860
590	Y(1,I) = SAVE(1,I)	LDA	4870
C	-----	LDA	4880
	CALL COPYZ (YL,YLSV,LCOPYL)	LDA	4890
	IWEVAL = 1	LDA	4900
	GC TO 200	LDA	4910
600	KFLAG = -5	LDA	4920
	GC TO 160	LDA	4930
C	-----	LDA	4940
C	THIS SECTION SCALES THE Y DERIVATIVES BY R**J	LDA	4950
C	-----	LDA	4960
610	R1 = 1.	LDA	4970
C	-----	LDA	4980
	DO 620 J=2,K	LDA	4990
C	R1 = R1*R	LDA	5000
C	-----	LDA	5010
	DO 620 I=1,NY	LDA	5020
620	Y(J,I) = Y(J,I)*R1	LDA	5030
C	-----	LDA	5040
	GC TO IRET, (190,510)	LDA	5050
C	-----	LDA	5060
C	THIS SECTION ALLOWS FOR RESTARTS AFTER SOLVING ANOTHER PROBLEM, OR	LDA	5070
C	HAVING TERMINATED THE CURRENT COMPUTER RUN. SUBROUTINE LDASAV	LDA	5080
C	SAVES THE NECESSARY VALUES WHICH ARE INTERNAL TO LCASLB. FOR	LDA	5090
C	DOUBLE PRECISION, WITH COPYZ IN SINGLE PRECISION, THE NUMBER OF	LDA	5100
C	LOCATIONS TO BE SAVED AND RESTORED, LCOPYS AND LCOPYR, MUST BE	LDA	5110
C	SET TO 56.	LDA	5120
C	IT IS ASSUMED THAT IN ADDITION TO THE VARIABLES IN THE ARRAY A	LDA	5130
C	SAVED BY CALLING LDASAV, THE USER ALSO SAVES THE ARRAYS SAVE,	LDA	5140
C	YLSV, YMAX, ESV, AND PW.	LDA	5150
C	-----	LDA	5160
C	TO RESTART THE USER FIRST CALLS LDARST TO RESTORE THE VALUES SAVED	LDA	5170
C	BY LDASAV, THEN RE-ENTERS LDASUB WITH JSTART < 0, AND WITH THE	LDA	5180
C	OTHER PARAMETERS THE SAME AS RETURNED FROM THE LAST ENTRY TO	LDA	5190
C	LDASUB, PARTICULARLY THOSE ARRAYS MENTIONED ABOVE.	LDA	5200
C	-----	LDA	5210
	ENTRY LCASAV(SAV)	LDA	5220
	LCOPYS = 29	LDA	5230
	CALL COPYZ (SAV,A,LCOPYS)	LDA	5240
	CALL COPYZ (SAVE,Y,LCOPYR)	LDA	5250
	CALL COPYZ (YLSV,YL,LCOPYL)	LDA	5260
	RETURN	LDA	5270
C	-----	LDA	5280
	ENTRY LCAARST(SAV)	LDA	5290
	LCOPYR = 29	LDA	5300
	CALL COPYZ (A,SAV,LCOPYR)	LDA	5310
	RETURN	LDA	5320
C	-----	LDA	5330
C	-----	LDA	5340
C	-----	LDA	5350
C	-----	LDA	5360

```

C
1  FCRMAT (2I5,I2,1D2E10.2,7E14.6/(32X,7E14.6))          LDA 5370
2  FCRMAT (32X,1F7E14.6)                                  LDA 5380
3  FCRMAT ('1' N='1,13,' NL='1,13,' RMSEPS='1PE5.2,' TEND=' LDA 5390
1  'E9.2,' F='E9.2//)                                     LDA 5400
4  FCRMAT (' NS NW Q H',8X,'T ',8X,'Y(1,*) AND YL(*)'//)   LDA 5410
   FNC                                                       LDA 5420
                                                           LDA 5430

SUBROUTINE CCOPYZ(S,Y,L)
DIMENSION S(1),Y(1)
-----
THIS SUBROUTINE COPIES THE ARRAY Y, OF LENGTH L, INTO THE ARRAY S
-----
IF(L.LE.0)RETURN
DO 100 J=1,L
S(J) = Y(J)
100 RETURN
ENC

SUBROUTINE CDEVAL (Y,YL,T,N,NY,W,KERET)
DER 10
DER 20
THIS SUBROUTINE CALCULATES THE INITIAL VALUES OF THE DERIVATIVES
DER 30
IN THE GENERAL CASE. IT IS WRITTEN SO THAT IT SHOULD WORK IF THE
DER 40
FIRST NY EQUATIONS ALL INVOLVE DERIVATIVES. IT ATTEMPTS TO SOLVE
DER 50
THE FIRST NY EQUATIONS USING NEWTON'S METHOD, BUT SINCE IT TRIES
DER 60
TO EVALUATE DF/DY BY CALLING JACMAT IN SUCH A WAY AS TO MAKE THE
DER 70
DF/DY TERM INSIGNIFICANT, IT IS POSSIBLE THAT IT MAY FAIL FOR THAT
DER 80
REASON. IT MAY FAIL FOR OTHER REASONS, AS WELL. IF IT DOES FAIL
DER 90
THE USER CAN SUPPLY HIS OWN VERSION OF DERVAL, OR MODIFY THIS
DER 100
ROUTINE IN SUITABLE FASHION. THIS ROUTINE ASSUMES THAT VALUES OF
DER 110
THE LINEAR VARIABLES HAVE BEEN SUPPLIED PREVIOUSLY. IF THOSE
DER 120
MUST BE SOLVED FOR SIMULTANEOUSLY WITH THE DERIVATIVES, THE USER
DER 130
MUST SUPPLY HIS OWN VERSION OF DERVAL.
DER 140
THE CALLING SEQUENCE FOR THIS SUBROUTINE IS
DER 150
CALL CDEVAL(Y,YL,T,N,NY,W,KERET)
DER 160
WHERE THE PARAMETERS ARE DEFINED AS FOLLOWS
DER 170
Y          - SAME AS IN LDASUB AND SDESOL. Y(1,I) CONTAINS THE
DER 180
INITIAL VALUES OF THE DEPENDENT VARIABLES. THE
DER 190
VALUES OF THE DERIVATIVES ARE RETURNED IN Y(2,I).
DER 200
YL         - SAME AS IN LDASUB AND SDESOL. THE INITIAL VALUES OF
DER 210
THE LINEAR VARIABLES MUST BE SUPPLIED TO THIS VERSION.
DER 220
T          - INITIAL TIME
DER 230
N          - SAME AS IN LDASUB, TOTAL NUMBER OF VARIABLES
DER 240
NY         - SAME AS IN LDASUB, NUMBER OF DIFFERENTIAL EQUATIONS
DER 250
AND NONLINEAR VARIABLES
DER 260
W          - SCRATCH ARRAY W FROM THE CALLING SEQUENCE OF SDESOL.
DER 270
THIS CAN BE USED AS NEEDED IN THIS SUBROUTINE.
DER 280
KERET      - RETURN INDICATOR
DER 290
              =0  NORMAL RETURN
DER 300
              =1  ERROR RETURN
DER 310
-----
C
DIMENSION Y(7,1), YL(1), W(1)
C
DO 100 I=1,NY
W(2*N+1) = MAX1(ABS(Y(1,I)),1.)
100 Y(3,I) = 0.
C
HINV = 1E-20
KERET = 0
EPS2 = NY/1.E8
EPS = SQRT(EPS2)
C
DO 140 IT=1,10
C
DO 110 I=1,NY
110 Y(2,I) = Y(3,I)/HINV

```

C	CALL DIFFUN (Y,YL,T,HINV,W)	DER	530
	CALL JACMAT (Y,YL,T,HINV,-1.,NY,NY,EPS,W,W(N+1),W(3*N+1))	DER	540
	NEWPW = 1	DER	550
C		DER	560
	CC 120 I=1,NY	DER	570
120	W(I) = W(I)*HINV	DER	580
C		DER	590
	CALL NUTSL (W(3*N+1),W,W(N+1),NY,NY,EPS,W(2*N+1),NEWPW,KRET)	DER	600
	IF (KRET.NE.0) GO TO 170	DER	610
	ER = 0.	DER	620
C		DER	630
	DC 130 I=1,NY	DER	640
	Y(3,I) = Y(2,I)-W(N+I)	DER	650
130	ER = ER+(W(N+I)/AMAX1(ABS(Y(3,I)),1.))*2	DER	660
C		DER	670
	IF (ER.LT.EPS2) GO TO 150	DER	680
140	CONTINUE	DER	690
C		DER	700
	GO TO 170	DER	710
C		DER	720
	CC 160 I=1,NY	DER	730
160	Y(2,I) = Y(3,I)	DER	740
C		DER	750
	RETURN	DER	760
170	KRET = 1	DER	770
	RETURN	DER	780
	END	DER	790
		DER	800

SUBROUTINE JACMAT (Y,YL,T,HINV,A2,N,NY,EPS,DY,F1,PW)

----- JAC 10
 JAC 20
 JAC 30
 SUBROUTINE JACMAT IS (USUALLY) SUPPLIED BY THE USER. ITS PURPOSE
 IS TO EVALUATE THE J MATRIX NEEDED WHEN THE CORRECTOR EQUATION
 IS SOLVED BY NEWTON'S METHOD. THIS VERSION APPROXIMATES
 J BY NUMERICAL DIFFERENCING AND USES FULL STORAGE MODE
 IN AN NXN MATRIX. JAC 40
 JAC 50
 JAC 60
 JAC 70
 JAC 80
 JAC 90

JACMAT CALCULATES THE MATRIX

$$J = \frac{DF}{CY} - \frac{A2}{F} \frac{DF}{DY}$$

THE CALLING SEQUENCE FOR THIS SUBROUTINE IS

CALL JACMAT(Y,YL,T,HINV,A2,N,NY,DY,F1,PW)
 WHERE THE PARAMETERS ARE DEFINED AS FOLLOWS.

Y	- SAME AS IN LDASUB AND IN SDESOL. ON INPUT TO THIS	JAC	220
	SUBROUTINE THE ARRAY CONTAINS CURRENT VALUES OF THE	JAC	230
	DEPENDENT VARIABLES AND THEIR (SCALED) DERIVATIVES.	JAC	240
YL	- SAME AS IN LDASUB AND IN SDESOL. ON INPUT TO THIS	JAC	250
	SUBROUTINE THE ARRAY CONTAINS CURRENT VALUES OF THE	JAC	260
	LINEAR VARIABLES.	JAC	270
T	- CURRENT TIME	JAC	280
HINV	- 1/H, WHERE H IS THE CURRENT STEPSIZE	JAC	290
A2	- A(2) FROM LDASUB.	JAC	300
N	- SAME AS IN LDASUB, TOTAL NUMBER OF VARIABLES	JAC	310
NY	- SAME AS IN LDASUB, NUMBER OF DIFFERENTIAL EQUATIONS	JAC	320
	AND NONLINEAR VARIABLES	JAC	330
EPS	- L2 ERROR CONSTANT USED IN LDASUB.	JAC	340
DY	- ARRAY OF FUNCTION VALUES AT CURRENT VALUES OF THE	JAC	350
	VARIABLES, INPUT TO JACMAT.	JAC	360
F1	- SCRATCH ARRAY OF N LOCATIONS WHICH CAN BE USED BY	JAC	370
	THIS SUBROUTINE IN ANY WAY NEEDED.	JAC	380
PW	- J MATRIX, OR APPROXIMATION, CALCULATED IN JACMAT AND	JAC	390
	RETURNED TO CALLING PROGRAM. THIS MATRIX IS USED IN	JAC	400
	SUBROUTINE NUTSL AND STORAGE MODE MUST AGREE BETWEEN	JAC	410
	THE TWO SUBROUTINES.	JAC	420
		JAC	430
		JAC	440
		JAC	450

 DIMENSION DY(1), Y(7,1), YL(1), F1(1), PW(1)

	AL = N-NY	JAC	460
	NN = N*N	JAC	470
C		JAC	480
	CC 100 I=1,NN	JAC	490
100	PW(I) = 0.	JAC	500
C		JAC	510
	CC 120 J=1,NY	JAC	520
	F = Y(1,J)	JAC	530
	E = Y(2,J)	JAC	540
	R = EPS*AMAX1(EPS,ABS(F),ABS(E))	JAC	550
	Y(1,J) = Y(1,J)+R	JAC	560
	Y(2,J) = Y(2,J)-A2*R	JAC	570
	CALL DIFFUN (Y,YL,T,HINV,F1)	JAC	580
C		JAC	590
	CC 110 I=1,N	JAC	600
110	PW(I+(J-1)*N) = (F1(I)-DY(I))/R	JAC	610
C		JAC	620
	Y(2,J) = F	JAC	630
120	Y(1,J) = F	JAC	640
C		JAC	650
	IF (NL.EC.0) GO TO 150	JAC	660
C		JAC	670
	CC 140 J=1,NL	JAC	680
	F = YL(J)	JAC	690
	R = EPS*AMAX1(EPS,ABS(F))	JAC	700
	YL(J) = YL(J)+R	JAC	710
	CALL DIFFUN (Y,YL,T,HINV,F1)	JAC	720
C		JAC	730
	CC 130 I=1,N	JAC	740
130	PW(I+(J+NY-1)*N) = (F1(I)-DY(I))/R	JAC	750
C		JAC	760
	140 YL(J) = F	JAC	770
C		JAC	780
	150 CONTINUE	JAC	790
	RETURN	JAC	800
	END	JAC	810
		JAC	820

	SUBROUTINE NUTSL (PW,DY,F1,N,NY,EPS,YMAX,NEWPW,KRET)	NUI	10
C	-----	NUI	20
C	THE PURPOSE OF THIS SUBROUTINE IS TO SOLVE A	NUI	30
C	LINEAR SYSTEM OF EQUATIONS FOR THE NEWTON ITERATES WHEN THE	NUI	40
C	CORRECTOR EQUATION IS BEING SOLVED. UPON ENTRY TO THIS SUBROUTINE	NUI	50
C	THE SYSTEM OF EQUATIONS TO BE SOLVED IS J W = -F, WHERE	NUI	60
C	J IS STORED IN PW UPON ENTRY	NUI	70
C	W IS RETURNED IN F1	NUI	80
C	-F IS STORED IN DY UPON ENTRY	NUI	90
C		NUI	100
C	THIS SUBROUTINE IS GENERALLY SUPPLIED BY THE USER, ALTHOUGH THERE	NUI	110
C	ARE SOME STANDARD FORMS AVAILABLE. FOR EXAMPLE, THIS VERSION	NUI	120
C	ASSUMES THAT PW IS STORED IN FULL STORAGE MODE IN AN NXN MATRIX.	NUI	130
C	IF NEWPW = 1, AN LU DECOMPOSITION IS DONE, NEWPW IS SET TO ZERO	NUI	140
C	AND FORWARD AND BACKWARD SUBSTITUTION FOR THE SOLUTION IS DONE.	NUI	150
C	IF NEWPW = 0, ONLY FORWARD AND BACKWARD SUBSTITUTION FOR THE	NUI	160
C	SOLUTION IS NECESSARY.	NUI	170
C		NUI	180
C	NOTE THAT THIS VERSION OF NUTSL REQUIRES THAT PW HAVE N**2 + 2*N	NUI	190
C	LOCATIONS SINCE 2*N LOCATIONS ARE USED BY THE IMSL LINEAR EQUATION	NUI	200
C	SOLVER.	NUI	210
C		NUI	220
C	NOTE THAT THE PARAMETERS EPS AND YMAX ARE USEFUL IF AN ITERATIVE	NUI	230
C	METHOD IS USED TO SOLVE THE SYSTEM OF EQUATIONS.	NUI	240
C		NUI	250
C	THE CALLING SEQUENCE FOR THIS SUBROUTINE IS	NUI	260
C	CALL NUTSL(PW,DY,F1,N,NY,EPS,YMAX,NEWPW,KRET)	NUI	270
C		NUI	280
C	WHERE THE PARAMETERS ARE DEFINED AS FOLLOWS.	NUI	290
C		NUI	300
C	PW - THE J MATRIX CALCULATED IN SUBROUTINE JACMAT	NUI	310
C	DY - THE RIGHT HAND SIDE OF THE LINEAR SYSTEM TO BE SOLVED	NUI	320
C	F1 - THE SOLUTION IS RETURNED IN THE ARRAY F1	NUI	330
C	N - SAME AS IN LDASUB, TOTAL NUMBER OF VARIABLES	NUI	340
C	NY - SAME AS IN LDASUB, NUMBER OF DIFFERENTIAL EQUATIONS	NUI	350
C		NUI	360


```

C      AND NONLINEAR VARIABLES
C      EPS      - L2 ERROR CONSTANT USED IN LDASUB
C      YMAX     - MAXIMUM VALUES OF Y(1,1) SEEN UP TO THE CURRENT TIME
C      NEWPW    - INDICATES WHETHER A NEW J MATRIX HAS BEEN COMPUTED
C                  =1 INDICATES A NEW J MATRIX HAS BEEN COMPUTED
C                  SINCE THE LAST ENTRY TO NUTSL. NEWPW
C                  SHOULD BE SET TO ZERO IF SOME PREPROCESSING
C                  SUCH AS LU DECOMPOSITION MUST BE DONE IN A
C                  NEW J MATRIX.
C                  =0 INDICATES THE J MATRIX IS THE SAME AS WHEN
C                  NUTSL WAS LAST ENTERED
C      KRET     - RETURN INDICATOR
C                  =0 NORMAL RETURN
C                  =1 ERROR RETURN. SOLUTION OF EQUATIONS COULD
C                  NOT BE OBTAINED.
C
C-----
C      DIMENSION PW(1), DY(1), F1(1), YMAX(1)
C      NL = N-NY
C      IF (NEWPW.EQ.0) GO TO 100
C      NEWPW = 0
C      NN = N**2+1
C      NNN = NN+N
C      CALL LUCATF (PW,PW,N,N,0,D1,D2,PW(NN),PW(NNN),F1,IER)
C      IF (IER.EQ.0) GO TO 100
C      KRET = 1
C      RETURN
100  CALL LUELMF (PW,DY,PW(NN),N,N,F1)
C      KRET = 0
C      RETURN
C      END

```

```

C      SUBROUTINE DIFFUN(Y,YL,T,HINV,DY)
C-----
C      SUBROUTINE DIFFUN IS SUPPLIED BY THE USER. ITS PURPOSE IS TO
C      EVALUATE THE FUNCTIONS AT CURRENT VALUES OF THE VARIABLES.
C
C      THE CALLING SEQUENCE FOR THIS SUBROUTINE IS
C
C      CALL DIFFUN(Y,YL,T,HINV,DY)
C
C      WHERE THE PARAMETERS ARE DEFINED AS FOLLOWS.
C
C      Y      - SAME AS IN LDASUB AND SDESOL. ON INPUT TO THIS
C              SUBROUTINE THE ARRAY CONTAINS CURRENT VALUES OF THE
C              DEPENDENT VARIABLES AND THEIR (SCALED) DERIVATIVES.
C      YL     - SAME AS IN LDASUB AND SDESOL. ON INPUT TO THIS
C              SUBROUTINE THE ARRAY CONTAINS CURRENT VALUES OF THE
C              LINEAR VARIABLES.
C      T      - CURRENT TIME
C      HINV   - 1/H, WHERE H IS THE CURRENT STEPSIZE
C      DY     - RETURNED ARRAY OF FUNCTION VALUES.
C
C-----
C      DIMENSION Y(7,1),YL(1),DY(1)
C      DEFINE YOUR FUNCTION HERE
C      RETURN
C      END

```


Appendix 2: Examples

Example 1: This example is the problem proposed by Gear [3]. The system of equations is

$$\dot{y}_i - S + (R - y_i)^2 + \sum_{j=1}^4 b_{ij} y_j = 0, \quad i = 1, 2, 3, 4$$

$$\text{where } R = \frac{1}{2} \sum_{i=1}^4 y_i \quad \text{and}$$

$$S = \frac{1}{2} \sum_{i=1}^4 (R - y_i)^2, \quad \text{and}$$

$$\dot{y}_5 + y_1 \dot{y}_6 + \dot{y}_1 y_6 = 0$$

$$2y_6 + y_6^3 - y_1 + v_1 - 1 - e^{-t} = 0$$

$$v_1 - v_2 + y_1 y_6 = 0$$

$$v_1 + v_2 + 5y_1 y_2 = 0.$$

$$\text{In the above } b_{11} = b_{22} = b_{33} = b_{44} = 447.501$$

$$b_{12} = -b_{34} = b_{21} = -b_{43} = -452.499$$

$$b_{13} = -b_{24} = b_{31} = -b_{42} = -47.499$$

$$b_{14} = -b_{23} = b_{41} = -b_{32} = -52.501.$$

The initial conditions are

$$y_i = 1, \quad i=1, 2, 3, 4.$$

$$y_5 = y_0 = 1$$

$$v_1 = -2, \quad v_2 = -3.$$

Note that a different version of DERVAL is necessary since $\frac{\partial F}{\partial \dot{y}}$ is singular.

```

DIMENSION Y(7,6),YL(2),W(150),GI(16)
COMMON /CAT/G(16)
DATA GI/447.501,-452.499,-47.499,-52.501,-452.499,447.501,52.501,
1 47.499,-47.499,52.501,447.501,452.499,-52.501,47.499,452.499,
2 447.501/
DATA N,NY,NL,M,REPS,HMAX,HMIN,H,T,TEND/8,6,2,6,1.E-3,5.E2,1.E-10,
1 1.E-4,0.2,1.E3/
CALL ERRSET(207,256,-1,1)
CALL ERRSET(208,256,-1,1)
DO 8 I=1,16
8 G(I) = GI(I)
DO 10 I=1,4
10 Y(1,I) = -1.
Y(1,5) = 1.
Y(1,6) = 1.
YL(1) = -2.
YL(2) = -3.
JSKF = 0
CALL SDESOL(Y,YL,T,TEND,NY,NL,M,JSKF,6,1,H,HMIN,HMAX,REPS,W)
PRINT 6,JSKF
6 FORMAT('C JSKF =',I4)
STOP
END

```

```

SUBROUTINE CIFFUN(Y,YL,T,HINV,DY)
COMMON /CAT/G(4,4)
DATA TOLC/-13.455/
DIMENSION Y(7,1),YL(1),DY(1)
IF(T.EQ.TCLD)GO TO 10
INTERM = EXP(-T)
TCLD = T
10 CONTINUE
S = (Y(1,1) + Y(1,2) + Y(1,3) + Y(1,4))/2.
S = 0.
DO 20 I=1,4
20 S = S + (R - Y(1,I))**2/2.
DO 30 I=1,4
30 DY(I) = HINV*Y(2,I) - S + (R - Y(1,I))**2
DO 25 J=1,4
25 DY(I) = DY(I) + G(I,J)*Y(1,J)
30 CONTINUE
DY(5) = HINV*(Y(2,5) + Y(1,1)*Y(2,6) + Y(2,1)*Y(1,6))
DY(6) = 2.*Y(1,6) + Y(1,6)**3 - Y(1,1) + YL(1) - 1.-INTERM
DY(7) = YL(1) - YL(2) + Y(1,1)*Y(1,6)
DY(8) = YL(1) + YL(2) + 5.*Y(1,1) * Y(1,2)
RETURN
END

```

```

SUBROUTINE Cerval(Y,YL,T,N,NY,W,KERET)
DIMENSION Y(7,1),YL(1),W(1)
KERET = 0
DO 50 I=1,NY
50 Y(2,I) = 0.
HINV = 1.
CALL CIFFUN(Y,YL,T,HINV,W)
DO 100 I=1,NY
100 Y(2,I) = -W(I)
RETURN
END

```

Example 2: This example is a small one, contrived to illustrate the possibility of derivatives entering in a nonlinear fashion. The equations are

$$\dot{y}_1 - 98y_1 + 98y_2 = 0$$

$$(\dot{y}_1)^2 + \dot{y}_2 - 198y_1 + e^{-t} y_1 + 199y_2 = 0$$

$$V_1 - y_1 - y_2 = 0$$

The initial conditions are

$$y_1 = y_2 = 1, V_1 = 2.$$

Note that we have supplied the explicit expression for the Jacobian. Either JACMAT or a modified version of DERVAL must be supplied as the numerical difference approximation to the Jacobian causes DERVAL to fail.

```

DIMENSION Y(7,2),YL(1),W(50)
DATA N,NY,NL,M,REPS,HMAX,HMIN,H,T,TEND/3,2,1,2,1.E-5,25.,1.E-10,
1 1.E-4,0.,50./
Y(1,1) = 1.
Y(1,2) = 1.
YL(1) = 2.
JSKF = 0
CALL SDESL(Y,YL,T,TEND,NY,NL,M,JSKF,6,1,F,HMIN,HMAX,REPS,W)
PRINT 6,JSKF
6 FCFMAT('0 JSKF =',I4)
STOP
END

```

```

SUBROUTINE CUFFUN(Y,YL,T,HINV,DY)
DIMENSION Y(7,1),YL(1),DY(1)
DATA TOLC/-79.03/
IF(T.EQ.TOLC)GO TO 10
TMTERM = EXP(-T)
TCLD = T
10 CONTINUE
DY(1) = Y(2,1)*HINV - 98.*Y(1,1) + 99.*Y(1,2)
DY(2) = (Y(2,1)*HINV)**2 + Y(2,2)*HINV - (198. - TMTERM)*Y(1,1) +
1 199.*Y(1,2)
DY(3) = YL(1) - Y(1,1) - Y(1,2)
RETURN
END

```

```

SUBROUTINE JACMAT(Y,YL,T,HINV,A2,N,NY,EPS,DY,F1,PW)
DIMENSION Y(7,1),YL(1),F1(1),DY(1),PW(N,1)
AH = A2*HINV
DO 100 I=1,N
DO 100 J=1,N
100 PW(I,J) = 0.
PW(1,1) = -AH - 98.
PW(1,2) = 99.
PW(2,1) = -2.*AH*Y(2,1)*HINV - 198. + EXP(-T)
PW(2,2) = -AH + 199.
IF(NL.LE.0)RETURN
PW(3,1) = -1.
PW(3,2) = -1.
PW(3,3) = 1.
RETURN
END

```

Example 3: This is another contrived example, this one to illustrate the use of one type of sparse matrix storage, along with the use of iteration to solve the equations (2.4). The system of equations is

$$\overline{A} \dot{y} + \overline{B} y = 0 ,$$

where

$$\overline{A} = \begin{pmatrix} 7 & 0 & 0 & -3 & 0 & -1 \\ 2 & 8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -3 & 0 & 0 & 5 & 0 & 0 \\ 0 & -1 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & -2 & 0 & 6 \end{pmatrix}$$

$$\overline{B} = \begin{pmatrix} .3 & 0 & 0 & .1 & 0 & -.2 \\ 1 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 10 & 0 & 0 & 20 & 0 & 0 \\ 0 & 5 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 57 & 0 & 100 \end{pmatrix}$$

The initial conditions are

$$y_1 = 1000$$

$$y_2 = 0$$

$$y_3 = -25$$

$$y_4 = 10$$

$$y_5 = 0$$

$$y_6 = -1000$$

The matrix storage scheme used for A, B, and the Jacobian, since it has nonzero elements in the same positions as A and B, is that outlined by Gustavson [4]. Briefly, one stores a pointer array (here called JS) which indicates the initial position of new elements in two other arrays, one of which (here called JN) gives the column number of the element stored in the corresponding position of the coefficient arrays (here called A and B).

Thus, for the above problem the arrays stored are

JS:	1	4	6	7	9	11	13					
	↓											
JN:	1	4	6	2	1	3	4	1	5	2	6	4
			→		→		→		→		→	
A :	7	-3	-1	8	2	1	5	-3	4	-1	6	-2
B :	.3	.1	-.2	3	1	1	20	10	6	5	100	57

The elements of the i^{th} row are stored beginning at location JS(i) of the array A, and in particular A(JS(i) + k) is the element in the i^{th} row and JN(JS(i) + k) th column of \bar{A} , for $k = 0, 1, \dots, \text{JS}(i+1) - \text{JS}(i) - 1$. For our purposes it is necessary to access the diagonal element easily, so we have required that the diagonal element be the first element stored for a given row. This means that $\text{JN}(\text{JS}(i)) = i$, $i=1, \dots, n$. Note that JS(i) is the number of nonzero elements in rows 1 through $i - 1$, and that JS(n+1) must be defined as the total number of nonzero elements.

Problems similar to the above arise when the finite element method is used to discretize the space domain for time dependent partial differential equations. Simple modifications to the subroutines given below should permit solution of large problems arising in that fashion. We

note, however, that it is not convenient to store symmetric matrices in this form unless all nonzero elements are stored. Storage of only the elements of the lower triangular matrix requires one to reference columns of the matrix, which are not readily accessible. Even if the entire matrix is stored, total storage requirements for matrices arising in finite element applications is still considerably less with this scheme than that required by symmetric band storage mode [5].

```

DIMENSION Y(7,6),W(126),YI(6),AD(12),BD(12),JSC(7),JND(12)
COMMON /DATA/A(12),B(12),N,JS(7),JN(12)
INTEGER*2 JS,JN
DATA 7,TEND,F,JSKF / 0.,250.,1.E-5,0 /
DATA JSC/1,4,6,7,9,11,13/
DATA JND/1,3,5,2,1,3,4,1,5,2,6,4/
DATA AD/7.,-3.,-1.,8.,2.,1.,5.,-3.,4.,-1.,6.,-2./
DATA BD/.3.,.1.,-.2,3.,1.,1.,20.,10.,3.,5.,100.,57./
DATA YI/1000.,0.,-25.,10.,3.,-1000./
N = 6
NP1 = N + 1
JE = JSC(NP1) - 1
DO 100 I=1,N
100 Y(1,I) = YI(I)
DO 110 I=1,JE
A(I) = AC(I)
B(I) = BC(I)
110 JN(I) = JND(I)
DO 120 I=1,NP1
120 JS(I) = JSC(I)
PRINT 4,N,(JS(I),I=1,NP1)
PRINT 5,(JN(I),I=1,JE)
PRINT 7,(A(I),I=1,JE)
PRINT 6,(B(I),I=1,JE)
CALL SDESOL(Y,YL,T,TEND,N,0,N,JSKF,6,1,H,1.E-6,125.,1.E-4,w)
PRINT 3,JSKF
STOP
3 FORMAT('O RETURN FROM SDESOL WITH JSKF =',I4)
4 FORMAT(' FOR THIS CASE N=',I3// ' THE JS ARRAY'/(12I10))
5 FORMAT('/O THE JN APRAY'/(12I10))
6 FORMAT('/O THE B ARRAY'/(12F10.2))
7 FORMAT('/O THE A ARRAY'/(12F10.2))
END

SUBROUTINE CIPFUN (Y,YL,T,HINV,DY)
COMMON /DATA/A(12),B(12),N,JS(7),JN(12)
INTEGER*2 JS,JN
DIMENSION Y(7,1),YL(1),DY(1)
DO 400 I=1,N
DY(I) = C.
JE = JS(I)
JE = JS(I+1) - 1
DO 300 J=JB,JE
300 DY(I) = DY(I) + Y(2,JN(J))*A(J)*HINV + B(J)*Y(1,JN(J))
400 CONTINUE
RETURN
END

SUBROUTINE NUTSL (PW,DY,F1,N,NY,EPS,YMAX,NEWPW,KRET)
COMMON /DATA/A(12),B(12),ND,JS(7),JN(12)
INTEGER*2 JS,JN
DIMENSION PW(1),DY(1),F1(1),YMAX(1)
DATA OMEG,CMEG/1 /1.05,.05/
KRET = 0
EPS = EPS**2
EPSA2 = EPS*.0001
NCIT = N
DO 100 I=1,NY
100 F1(I) = DY(I)/PW(JS(I))
DO 300 IT=1,NCIT
RCH = 0.
CH = 0.
DO 200 I=1,NY
JB = JS(I) + 1
JE = JS(I+1) - 1
FN = DY(I)
IF (JB.GT.JE) GO TO 180
DO 150 J=JB,JE
150 FN = FN - PW(J)*F1(JN(J))
180 FN = FN/PW(JE-1)
FN = FN*CMEG - F1(I)*GMEGM1
ACH = F1(I) - FN

```

```

      CH = CH + (ACH/YMAX(I))**2
      RCH = RCH + (ACH/AMAX1(ABS(FN),EPS))**2
200  F1(I) = FN
      IF(RCH.LT.EPSS) RETURN
      IF(CH.LT.EPSA2) RETURN
300  CCNTINUE
      KRET = 1
      RETURN
      END

```

```

      SUBROUTINE JACMAT(Y,YL,T,HINV,A2,N,NY,EPS,DY,F1,PW)
      COMMON /DATA/A(12),B(12),NDUM,JS(7),JN(12)
      INTEGER*2 JS,JN
      DIMENSION Y(7,1),YL(1),F1(1),DY(1),PW(1)
      AH = -A2*HINV
      JE = JS(N+1) - 1
      DO 100 J=1,JE
100  PW(J) = AH*A(J) + B(J)
      RETURN
      END

```

Example 4: This example arises from a nonlinear reactor dynamics problem where the finite element method is used to discretize the space domain. The resulting system of equations has the form

$$\overline{A} \dot{y} - \overline{B} y + w(\overline{C} y) y = 0 ,$$

where \overline{C} is a matrix with three subscripts. The i^{th} equation can be expressed as

$$\sum_{j=1}^N (\overline{a}_{ij} \dot{y}_j - \overline{b}_{ij} y_j) + w \sum_{j=1}^N \sum_{k=1}^N \overline{c}_{ijk} y_j y_k = 0 .$$

In this example $N = 28$, and there are at most seven nonzero elements per row in \overline{A} and \overline{B} . The nonlinear term $y_j y_k$ appears only if both \overline{a}_{ij} and \overline{a}_{ik} are nonzero. Therefore a different type of sparse matrix storage is used for this problem.

An array, K , dimensioned $(28, 7)$ is used to store (for each row), the columns subscripts for the nonzero elements. For convenience in accessing the diagonal element, we require that $K(i,1) = i$. We can note this matrix is simply the connectivity matrix for the finite element grid. Then the nonzero elements of \overline{A} and \overline{B} , are stored in the corresponding portions of the arrays A and B respectively. If there are in fact less than seven nonzero coefficients in a row, the remaining $K(i,j)$ are set to zero.

The storage for \overline{C} is somewhat more complicated. \overline{C} is symmetric (invariant under any permutation of subscripts). The nonlinear term of the i^{th} equation was rewritten as

$$w \sum_{j=1}^N \sum_{k=1}^N \bar{c}_{ijk} y_j y_k = w \sum_{j=1}^N \sum_{k=j}^N \bar{d}_{ijk} y_j y_k ,$$

where

$$\bar{d}_{ijk} = \begin{cases} 0 & k < j \\ \bar{c}_{ijk} & k = j \\ \bar{c}_{ijk} + c_{ikj} , & k > j . \end{cases}$$

The coefficients \bar{d}_{ijk} are then stored in a (28,28) array C in the order the second and third subscripts are given here.

(K(i,1), K(i,1)), ..., (K(i,1), K(i,7)), (K(i,2), K(i,2)), ..., (K(i,2), K(i,7)), ..., (K(i,7), K(i,7)) .

The equations can then be written in the form

$$\sum_{j=1}^7 [A_{ij} \dot{y}_{K(i,j)} - B_{ij} y_{K(i,j)}] + \sum_{j=i}^7 \sum_{k=j}^7 C_{im_{jk}} y_{K(i,j)} y_{K(i,k)} = 0$$

where

$$m_{jk} = k + \frac{(j-1)(14-j)}{2} .$$

Because of the large amount of data for this problem the input arrays K , A , B , and C are simply listed along with the programs for this example.


```

DIMENSION Y(7,28),WS(560)
COMMON /DATA/A(28,7),B(28,7),C(28,28),N,NNZ,K(28,7)
INTEGER*2 K
DATA TFND,FMIN,HMAX,EPS,ZOMEGA/.1,1.E-12,.1,.01,650.903E-14/
DATA NN,NY,NL/28,28,0/
NNZ = 7
CALL EPRSET(207,256,-1,1)
CALL ERRSET(208,256,-1,1)
N = NN
M = N
NEAC = NNZ*(NNZ + 1)/2
PRINT 10
CC 110 I=1,NN
READ 1,(K(I,J),J=1,NNZ)
110 PRINT 11,(K(I,J),J=1,NNZ)
PPINT 12
CC 120 I=1,NN
READ 2,(A(I,J),J=1,NNZ)
Y(1,I) = 0.
120 PRINT 15,(A(I,J),J=1,NNZ)
Y(1,1) = 1.E16
PRINT 13
CC 130 I=1,NN
READ 2,(B(I,J),J=1,NNZ)
130 PRINT 15,(B(I,J),J=1,NNZ)
PRINT 14
CC 140 I=1,NN
READ 2,(C(I,J),J=1,NEND)
CC 155 J=1,NEND
135 C(I,J) = C(I,J) *ZOMEGA
140 PRINT 15,(C(I,J),J=1,NEND)
JSKF = 0
T = 0.
H = HMIN*1000.
CALL SDESOL(Y,YL,T,TEND,NY,NL,M,JSKF,6,1,F,FMIN,HMAX,EPS,WS)
PRINT 3,JSKF
STOP
1 FORMAT(16I5)
2 FORMAT(7E11.4)
3 FORMAT('O JSKF = ',I3)
10 FORMAT('1K ARRAY')
11 FORMAT(8X,14I8)
12 FORMAT('///'0A(I,J)')
13 FORMAT('///'0B(I,J)')
14 FORMAT('///'0C(I,J)')
15 FORMAT(8X,1P7E16.6)
END

SUBROUTINE CIFFUN(Y,YL,T,HINV,DY)
COMMON /DATA/A(28,7),B(28,7),C(28,28),N,NNZ,K(28,7)
INTEGER*2 K
DIMENSION Y(7,1),YL(1),DY(1)
DO 400 I=1,N
CY(I) = 0.
DO 300 J=1,NNZ
IF(K(I,J).LE.0)GO TO 310
DY(I) = CY(I) + Y(2,K(I,J))*A(I,J)*HINV - B(I,J)*Y(1,K(I,J)) +
1 C(I,J)*Y(1,K(I,J))*Y(1,K(I,1))
300 CCNTINUE
310 L = NNZ
CC 360 J1=2,NNZ
IF(K(I,J1).LE.0)GO TO 400
DC 350 J2=J1,NNZ
L = L + 1
IF(K(I,J2).LE.0)GO TO 350
DY(I) = CY(I) + C(I,L)*Y(1,K(I,J1))*Y(1,K(I,J2))
350 CCNTINUE
360 CCNTINUE
400 CCNTINUE
RETURN
END

SUBROUTINE JACMAT(Y,YL,T,HINV,A2,N,NY,EPS,CY,F1,PW)

```

```

COMMON /DATA/A(28,7),B(28,7),C(28,28),ND,NNZ,K(28,7)
INTEGER*2 K,P
DIMENSION Y(7,1),YL(1),F1(1),DY(1),PW(NY,1)
DIMENSION P(7,7)
DATA P/45*0/,INITP/0/
IF (INITP.EC.NNZ)GO TO 99
INITP = NNZ
DO 98 L=1,NNZ
DO 98 M = L,NNZ
98 P(L,M) = M + (L - 1)*(2*NNZ - L)/2
99 CCNTINUE
AH = -A2*FINV
DO 300 I=1,NY
DO 300 J=1,NNZ
PW(I,J) = AH*A(I,J) - B(I,J)
DO 100 L=1,J
IF(K(I,L).LE.0)GO TO 295
100 PW(I,J) = PW(I,J) + C(I,P(L,J))*Y(I,K(I,L))
DO 200 M=J,NNZ
IF(K(I,M).LE.0)GO TO 295
200 PW(I,J) = PW(I,J) + C(I,P(J,M))*Y(I,K(I,M))
295 CCNTINUE
300 CCNTINUE
RETURN
END

```

```

SUBROUTINE NUTISL(PW,DY,F1,N,NY,EPS,YMAX,NEWPW,KRET)
DIMENSION PW(NY,1),DY(1),F1(1),YMAX(1)
COMMON /DATA/A(28,7),B(28,7),C(28,28),ND,NNZ,K(28,7)
DATA SPC,SPDM1/1.05,.05/
INTEGER*2 K
KRET = 0
EPSS = EPS**2
EPSA2 = EPSS*.0001
NCIT = N
280 DO 281 I=1,NY
281 F1(I) = CY(I)/PW(I,1)
DO 287 IT=1,NCIT
RCH = 0.
CH = 0.
DO 285 I=1,NY
FN = DY(I)
DO 284 J=2,NNZ
IF(K(I,J).LE.0.OP.K(I,J).GT.NY)GO TO 284
FN = FN - PW(I,J)*F1(K(I,J))
284 CCNTINUE
FN = FN/PW(I,1)
FN = FN*SPC - SPDM1*F1(I)
ACH = F1(I) - FN
CH = CH + (ACH/YMAX(I))**2
RCH = RCH + (ACH/AMAX1(ABS(FN),EPS))**2
285 F1(I) = FN
IF(RCH.LT.EPSS)GO TO 288
IF(CH.LE.EPSA2)GO TO 288
287 CCNTINUE
KRET = 3
288 CCNTINUE
RETURN
END

```

INPUT DATA FOR EXAMPLE 4

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	8												

-7.2321E	10	1.5500E	10	3.1768E	09	1.3103E	10	1.1476E	10	1.1476E	10	1.6289E	10
-8.2448E	10	1.7989E	10	1.3103E	10	2.4060E	10	3.6945E	08	1.6043E	10	0.0	10
-8.7115E	09	1.0212E	10	3.1768E	09	1.6043E	09	2.4798E	10	3.4658E	09	1.3396E	10
-4.7063E	10	2.8953E	09	6.0318E	08	2.8953E	09	6.4005E	08				
-1.0116E	11	6.4005E	08	8.4446E	09	2.0409E	10	4.6822E	09				
-1.3320E	11	1.1621E	10	3.4658E	09	2.3750E	10						
-1.3320E	11	2.4768E	10	3.6945E	08	7.2533E	09	2.3750E	10				
-1.4888E	11	1.1476E	10	1.1476E	10	3.5688E	09						
-7.4644E	10	1.1476E	10	2.4060E	10	3.3923E	09	3.5688E	09				
4.1888E	02	5.5850E	02	2.7925E	02	0.0	02	0.0		0.0		0.0	
4.1888E	02	2.7925E	02	0.0	02	0.0		0.0		0.0		1.3963E	02
0.0		0.0		0.0		0.0		0.0		0.0		0.0	
0.0		0.0		0.0		0.0		0.0		0.0		0.0	
2.0		0.0		8.3776E	02	0.0		5.5850E	02	0.0		0.0	
1.3963E	03	2.7925E	02	0.0		0.0		0.0		0.0		0.0	
0.0		0.0		0.0		0.0		0.0		0.0		0.0	
2.2340E	03	9.7738E	02	0.0		5.5850E	02	1.1170E	03	5.5850E	02	0.0	
2.2340E	03	9.7738E	02	0.0		8.3776E	02	1.3963E	03	5.5850E	02	0.0	
1.3963E	03	2.7925E	02	0.0		0.0		0.0		0.0		0.0	
2.2340E	03	0.0		0.0		0.0		1.3963E	03	0.0		0.0	
0.0		0.0		0.0		0.0		0.0		0.0		0.0	
2.0		0.0		8.3776E	02	0.0		5.5850E	02	0.0		0.0	
1.3963E	03	2.7925E	02	0.0		0.0		0.0		0.0		0.0	
0.0		0.0		0.0		0.0		0.0		0.0		0.0	
2.2340E	03	9.7738E	02	0.0		5.5850E	02	1.1170E	03	5.5850E	02	0.0	
4.1888E	02	2.7925E	02	0.0		8.3776E	02	1.3963E	03	5.5850E	02	0.0	
0.0		0.0		0.0		0.0		0.0		0.0		0.0	
0.0		0.0		0.0		0.0		0.0		0.0		0.0	
6.7021E	03	2.2340E	03	1.9548E	03	8.3776E	02	0.0		0.0		0.0	
1.2556E	03	1.1170E	03	0.0		0.0		0.0		0.0		0.0	
5.5850E	02	0.0		0.0		1.3963E	03	6.9813E	02	0.0		0.0	
1.1170E	03	2.7925E	02	0.0		1.1170E	03	0.0		0.0		0.0	
1.6755E	03	2.2340E	03	1.3963E	03	2.5133E	03	0.0		2.2340E	03	1.3963E	03
8.3776E	02	5.5850E	02	0.0		0.0		0.0		0.0		1.6755E	03
1.1170E	03	0.0		0.0		1.9548E	03	2.6529E	03	0.0		0.0	
2.2340E	03	2.7925E	03	1.3963E	03	6.4228E	03	2.6529E	03	2.5133E	03	5.3058E	03
1.5368E	02	0.0		1.1170E	03	1.9548E	03	4.4680E	03	1.9548E	03	8.3776E	02
0.0		0.0		0.0		0.0		0.0		1.9548E	03	6.9813E	02
1.5548E	03	2.5133E	03	1.1170E	03	8.3776E	02	1.6755E	03	1.1170E	03	0.0	
1.6755E	04	2.2340E	03	1.3963E	03	0.0		1.6755E	03	2.2340E	03	1.6151E	03
8.3776E	02	5.5850E	02	0.0		0.0		0.0		0.0		1.3963E	03
1.1170E	03	0.0		0.0		1.9543E	03	0.0		0.0		1.6755E	03
2.2340E	03	2.7925E	03	1.3963E	03	6.4228E	03	2.6529E	03	2.5133E	03	5.3058E	03
6.7021E	03	0.0		1.1170E	03	1.9548E	03	2.2340E	03	0.0		0.0	
2.7925E	02	2.7925E	02	0.0		0.0		0.0		8.3776E	02	6.9813E	02
0.0		0.0		0.0		8.3776E	02	1.6755E	03	1.1170E	03	0.0	
1.5548E	03	1.2566E	03	0.0		0.0		0.0		0.0		0.0	
1.4242E	04	0.0		3.9095E	03	3.6303E	03	2.5133E	03	0.0		0.0	
1.1170E	03	0.0		1.1170E	03	1.1170E	03	0.0		2.5133E	03	2.0944E	03
1.5548E	03	0.0		0.0		0.0		0.0		1.3963E	03	0.0	
3.0718E	03	2.5133E	03	0.0		2.7925E	03	3.3510E	03	0.0		0.0	
3.1833E	04	5.5850E	03	3.0718E	03	4.1888E	03	0.0		3.9095E	03	3.0718E	03
2.5133E	03	1.3963E	03	0.0		0.0		0.0		0.0		3.3510E	03
1.5548E	03	0.0		0.0		3.6303E	03	4.3284E	03	0.0		0.0	
3.9095E	03	4.4680E	03	2.2340E	03	1.0612E	04	4.3284E	03	4.1888E	03	1.0332E	04
2.8484E	04	2.7925E	03	2.7925E	03	3.6303E	03	7.8151E	03	3.6303E	03	2.5133E	03
0.0		1.1170E	03	0.0		0.0		0.0		6.1436E	03	2.3736E	03
4.4680E	03	0.0		0.0		2.5133E	03	3.3510E	03	1.9548E	03	0.0	
3.1833E	04	5.5850E	03	1.9548E	03	0.0		3.3510E	03	4.4680E	03	5.1662E	03
2.5133E	03	1.3963E	03	0.0		0.0		0.0		3.9095E	03	3.0718E	03
1.5548E	03	0.0		0.0		3.6303E	02	4.3284E	03	0.0		3.3510E	03
3.9095E	03	4.4680E	03	2.2340E	03	1.0612E	04	4.3284E	03	4.1888E	03	1.0332E	04
1.4242E	04	0.0		2.7925E	03	3.6303E	03	3.9095E	03	0.0		0.0	
1.1170E	03	1.1170E	03	0.0		0.0		0.0		2.5133E	03	2.3736E	03
0.0		0.0		0.0		2.5133E	03	3.3510E	03	1.9548E	03	0.0	
4.4680E	03	0.0		0.0		0.0		0.0		0.0		0.0	
1.3822E	04	0.0		0.0		0.0		4.1888E	03	0.0		0.0	
1.5548E	03	0.0		0.0		1.9548E	03	0.0		4.1888E	03	0.0	
0.0		0.0		0.0		0.0		2.3736E	03	2.2340E	03	0.0	
4.7473E	03	4.0452E	03	0.0		4.4680E	03	0.0		0.0		0.0	

50

References

1. R. L. Brown and C. W. Gear, "Documentation for DFASUB - a program for the solution of simultaneous implicit differential and nonlinear equations," Report no. UIUCDCS-R-73-575, University of Illinois at Urbana - Champaign, Urbana, Illinois, July 1973.
2. C. W. Gear Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, Englewood Cliffs, NJ, 1971.
3. C. W. Gear, "Simultaneous Numerical Solution of Differential-Algebraic Systems," IEEE Trans. on Circuit Theory, CT-18(1971)89-95.
4. F. G. Gustavson, "Some Basic Techniques for Solving Sparse Systems of Linear Equations," pp 41-52 in Sparse Matrices and Their Applications, D. J. Rose and R. A. Willoughby, eds., Plenum Press, New York - London, 1972.
5. D. Salinas, D. H. Nguyen, R. Olsen, and R. Franke "An Optimal Compact Storage Scheme for Nonlinear Reactor Problems by FEM," Manuscript (to be submitted).

Distribution List

	<u>No. of copies</u>
Defense Documentation Center Cameron Station Alexandria, VA 22314	12
Library Naval Postgraduate School Monterey, CA 93940	2
Dean of Research Naval Postgraduate School Monterey, CA 93940	2
Naval Postgraduate School Department of Mathematics Ladis D. Kovach, Chairman	1
Professor C. Comstock	1
Professor F. Faulkner	1
Professor R. Franke	10
Naval Postgraduate School Department of Mechanical Engineering Professor D. Salinas	1
Professor D. Nguyen	1
Professor R. Newton	1
Professor G. Cantin	1
Naval Postgraduate School Department of Aeronautics Professor D. Collins	1
Professor R. Ball	1
Naval Postgraduate School Computer Center Monterey, CA 93940 Professor D. Williams	1
Mr. Roger Hilleary	1
Dr. Richard Lau Office of Naval Research Pasadena, CA 91100	1
Chief of Naval Research ATTN: Mathematics Program Arlington, VA 22217	2

Mr. W. J. Dejka, Code 4000 Naval Electronics Laboratory Center San Diego, CA 92152	1
Argonne National Laboratory Argonne, IL 60439	
ATTN: Mr. Gary Leaf, AMD	1
Mr. Tilak Chawla, RAS	1
Professor C. W. Gear University of Illinois Urbana, IL 61801	1
Dr. A. C. Hindmarsh Lawrence Livermore Laboratory Livermore, CA 94550	1
Sandia Laboratories Albuquerque, NM 87115	
ATTN: D. A. Dahlgren	1
L. F. Shampine	1
Mr. R. E. Huddleston Sandia Laboratories Livermore, CA 94550	1
Air Force Weapons Laboratory Kirtland AFB Albuquerque, NM 87115	
ATTN: C. M. Walters	1
CAPT. C. W. Stein	
Mr. R. D. Birkhoff Oak Ridge National Laboratory Union Carbide Corporation P. O. Box X Oak Ridge, TN 37830	1
Mr. R. M. Sternheimer Brookhaven National Laboratory Upton, NY 11973	1
Mr. B. F. Maskewitz RSIC, Neutron Physics Division Oak Ridge National Laboratory Oak Ridge, TN 37831	1
Mr. J. L. Black U. S. Naval Research Laboratory Code 770 Washington, DC 20390	1

Los Alamos Scientific Laboratory
P. O. Box 1663
Los Alamos, NM 87544
ATTN: S. Evans, T-6
C. Young, J-14

1
1

Professor R. E. Barnhill
Department of Mathematics
University of Utah
Salt Lake City, Utah 84112

1

LT Donald Hinsman
Fleet Numerical Weather Central
Monterey, CA 93940

1

U173475

DUDLEY KNOX LIBRARY - RESEARCH REPORTS



5 6853 01071186 4

NPS